1994年9月1日発行（毎月1回1日発行）第13巻9号通巻149号 昭和58年11月2日第三種郵便物認可

PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

特集 SX-WINDOW環境セットアップ

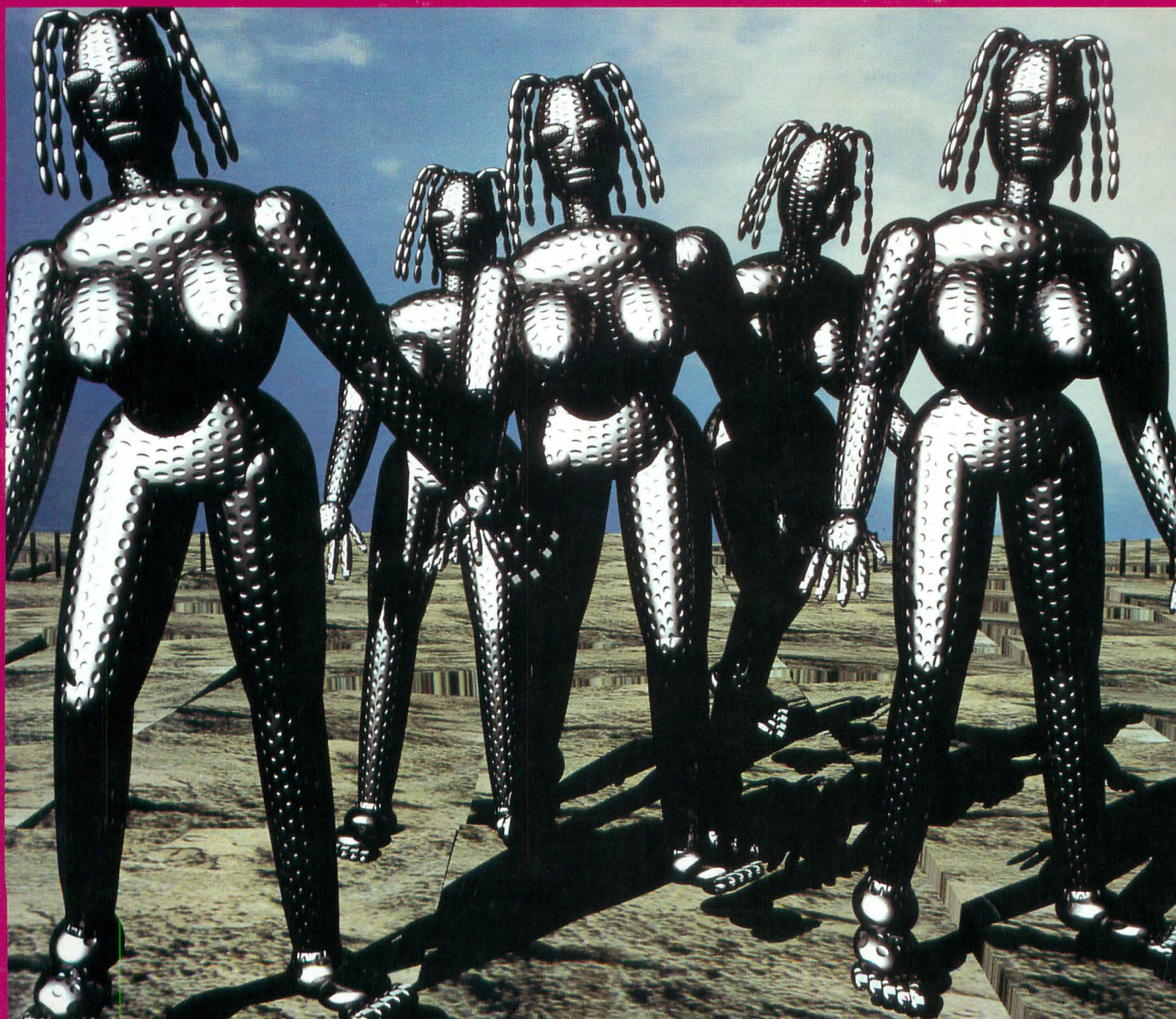
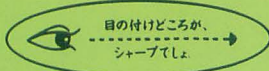
シャーペンをカスタマイズしよう/外部コマンドを作成する/SYSDTOP.SXを斬る

新製品MJ-700V2C/X68030 D'ash/新刊X680x0 TEX

「PUSH BON!」オリジナルステージデータ50面

1994

SOFT BANK オー！エックス

定価680円

■実画面：1,024×1,024ドット、表示画：768×：512ドット

●画面は広告用に作成した、機能を説明するためのイメージ画面です。また、各種アイコンなどは、SX-WINDOW ver.3.1がもつ機能を使って作成したもので、標準装備のものとは異なるものもあります。
●本広告中の「シャーペン」で表示している文字のフォントはツァイト社の、「書体倶楽部」のフォントを使用しています。

❶「パターンエディタ」で作成したデータを背景に設定可能。

❷日本語フロントプロセッサ ASK68K ver.3.0の辞書メンテナンスがウィンドウ上で可能。

❸ESC/Page、LIPSIII、PostScriptに対応したプリンタが利用できます。

❹付属アプリケーション「シャーペン」編集例。文字ごとに文字種・文字の大きさの指定、装飾が可能。またインライン入力をサポート、イメージデータの貼りつけもOK。

❺512×512ドットの範囲内で65,536色の表示が可能。

❻「CGAウィンドウ」、65,536色（最大）のコンピュータアニメーション表示が可能。

❼異なる画像フォーマットへのコンバートが可能。

❽アイコンデータや背景データを作成する「パターンエディタ」。

❾オリジナルに作成したアイコンパターンの例。

❿Human68kやX-BASICのコマンドをSX-WINDOWアプリケーションと同時にタイムシェアリングで実行できます。

■「SX-WINDOW ver3.1システムキット」のお問い合わせは…シャープ㈱電子機器事業本部（液映）システム機器推進プロジェクトチーム　〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161（大代表）へ

C　O　N　T

〈スタッフ〉
●編集長／前田　徹　●副編集長／植木章夫　●編集／山田純二　豊浦史子　高橋恒行　●協力／有田隆也　中森　章　林　一樹　吉田幸一　華門真人　朝倉祐二　大和　哲　村田敏幸　丹　明彦　三沢和彦　長沢淳博　司馬　護　清瀬栄介　石上達也　柴田　淳　瀧　康史　横内威至　進藤慶到　●カメラ／杉山和美　●イラスト／山田晴久　江口響子　高橋哲史　川原由唯　●アートディレクター／島村勝頼　●レイアウト／元木昌子　ADGREEN　●校正／グループごじら

特集　SX-WINDOW環境セットアップ

餓狼伝説SPECIAL

THE USER'S WORKS

オリジナルステージデータ50面

MJ-700V2C

(で)のショートプロぱーてぃ

表紙絵：塚田 哲也

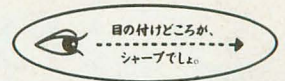
E N T S

1994 SEP.

9

●連載/紹介/講座/プログラム

■広告目次

ビデオグラフィックスの
世界へ。
資料請求券
X68030
Oh!X
9係
■お問い合わせは… シャープ株式会社
電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

1,677万色対応、ビデオ映像を高画質・高速取り込み

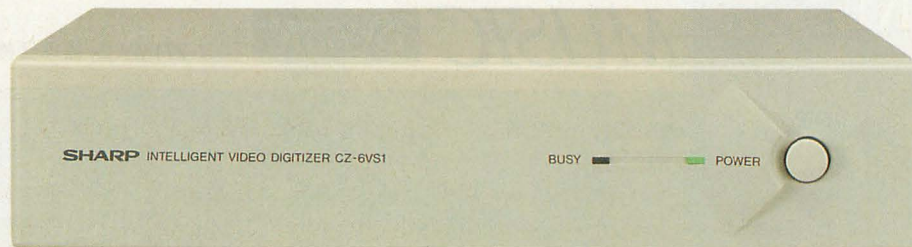
テレビやビデオ、ビデオディスクなどの映像をX68シリーズやMacシリーズ※1の動画・静止画データとして高速取り込みが可能、いわば"ビデオスキャナ"とでも呼びたいビデオ入力ユニットです。1,677万色対応、最大640×480ドットの高解像度※2。動画・静止画の手軽なハンドリングが、新たなグラフィックシーンを創造します。

※1 MacintoshはIIシリーズ以降の機種に対応、ディスプレイ解像度が640×480ドットの場合、取り込み可能な範囲は、160×120ドット、320×240ドットのサイズになります。
※2 X68030/X68000シリーズでは、1,677万色はデータ作成のみに対応。表示は最大65,536色、解像度は512×512ドット。また、Macintoshは機種により表示色数が異なります。

アプリケーションツール「ライブスキャン」を標準装備

動画や静止画を簡単に保存できるアプリケーションソフト「ライブスキャン」※を標準装備。取り込んでいる映像を表示したり、残したいシーンを簡単に静止画保存したり、手軽な動画・静止画ハンドリングでパソコンの可能性をさらに広げます。X68030/X68000シリーズ用SX-WINDOW対応版とMacintoshシリーズ用QuickTime対応版の2種類を同梱しています。

※SX-WINDOW版はバージョン3.0以降(メモリー4MB以上)、QuickTime版はMacintosh漢字Talk7リリース7.1以上のシステムとQuickTime1.5以上(メモリー8MB以上)が必要です。

1,677万色対応の高速映像取り込み、動画・静止画の手軽なハンドリングが、新たなマルチメディアシーンを創造する。

■SCSIインターフェイス採用:パソコンの専用I/Oスロットを使わずに接続可能になり、汎用化を実現しました。またSCSI-2(FAST)インターフェイスの採用により、データ転送速度の高速化を図っています。X68030/X68000シリーズでは、SCSI-2(FAST)対応のハードディスクを接続することにより、パソコン本体を経由しないで、ハードディスクに直接、動画データをテンポラリデータとして記録することが可能です。パソコン本体のハードディスクへは、記録終了後に、テンポラリデータを変換し動画データとして保存できます。

※CZ-600C/601C/611C/602C/612C/652C/662C/603C/613C/653C/663Cに接続する場合は別売のSCSIインターフェイスボードCZ-6BS1ならびにSCSI変換ケーブルCZ-6CS1が必要です。※CZ-604C/623C/634C/644Cに接続する場合は、別売のSCSI変換ケーブルCZ-6CS1が必要です。
※Macintosh Power Bookシリーズに接続する場合は別売のSCSIケーブルなどが必要です。詳しくはMacintosh Power Bookシリーズの取扱説明書をご覧ください。

■高機能MPUを搭載:クロック周波数25MHzの32ビットMPU/MC68EC020を搭載、高速処理やパソコン本体の負担の軽減を実現します。

●MacはMacintoshの略称です。●Macintosh、Macintosh IIは、米国アップルコンピュータ社の登録商標です。●Power Bookは米国アップルコンピュータ社の商標です。●漢字Talk7はアップルコンピュータジャパン社の商標です。●QuickTimeは、米国アップルコンピュータ社の商標です。●価格には、消費税及び配送・設置・付帯工事費、使用済み商品の引き取り費等は含まれておりません。

●さらに実用的なウィンドウシステムへの進化

SX-WINDOW ver3.1 システムキット

CZ-296SS(130㎜FD)/CZ-296SSC(90㎜FD) 標準価格22,800円(税別) NEW

ASK68K Ver3.0を利用したインライン入力のサポート、Human68k/BASICコマンドをSX-WINDOWアプリケーションと同時にタイムシェアリングで実行できるコンソールのサポートをはじめ、シャーペン.Xをワープロとして利用できるよう機能アップ。また、さまざまなSX-WINDOWアプリケーションで利用できるページプリンタドライバを標準装備。ドローデータ(FSX)/フォントデータ(IFM)処理の高速化も実現しています。

※コンソールでは、SX-WINDOWと処理が重複するものは実行できません。 4MB

※既にSX-WINDOWをお持ちの方には有償バージョンアップサービスを行います。

●定評のGUI対応ウィンドウワープロ

EGWord SX-68K

CZ-271BWD 標準価格59,800円(税別) NEW

ウィンドウワープロとして評価の高いEGWordのSX-WINDOW対応版。キャラクタベースのワープロを超えたグラフィカルユーザーインターフェイス(GUI)による手軽なDTPソフトとしても優れた表現力を発揮します。定評ある日本語入力方式(EGConvert)によるインライン入力、さまざまなグラフィックデータ(GScript)やテキストデータの貼り込み、また文書互換を実現するEDF(Extended Document Format)形式をサポートしています。 4MB、ver.2.0

●SX-WINDOW開発支援ツール

SX-WINDOW 開発キット Workroom SX-68K

CZ-288LWD 標準価格39,800円(税別) NEW

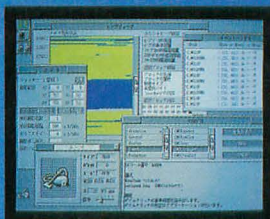
SX-WINDOW用のソフト開発に必要なツールやサンプルプログラムを装備。プログラムの編集、リソースの作成、コンパイル、デバッグといった一連の作業をSX-WINDOW上で効率よく実行できます。初めてSX-WINDOW用のプログラムに挑戦する人にも、簡単に基本機能の理解が深まる33種(基礎編23種、応用編4種、実用編6種)のサンプルプログラム付き。

※ご使用に当ってはC compiler PRO-68K ver.2.1が必要です。 4MB、ver.2.0

●SX-WINDOW開発キットのサポートツール

開発キット用ツール集

CZ-289TWD 標準価格12,800円(税別) NEW

SX-WINDOW開発キットをさらに使いやすくするためのツールです。SXコールの簡易リファレンスを簡単に検索するインサイドSX、イベントの発生を常時監視・確認するイベントハンドラ、リアルタイムにメモリブロックの利用状況を表示するヒープビューアなど11種のツールが用意されています。 2MB、ver.2.0

●SX-WINDOW対応ドローイングツール

Easydraw SX-68K

CZ-264GWD 標準価格19,800円(税別)

イラスト、フローチャート、地図、見取り図など各種グラフィックが製図感覚で作成できます。作成したデータは他のSX-WINDOW対応アプリケーションでも利用でき、企画書などの作成をサポート。ページプリンタドライバも標準装備。 4MB、ver.3.0

●ウィンドウ対応グラフィックツール

Easypaint SX-68K

CZ-263GWD 標準価格12,800円(税別)

マウスによる簡単操作、65,536色中16色の多彩な表現、クリエイティブマインドに応えるウィンドウ対応ペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間でのデータ交換もできます。 2MB、ver.1.1

●SX-WINDOWを楽しく使うためのアクセサリ集

SX-WINDOW デスクアクセサリ集

CZ-290TWD 標準価格14,800円(税別)

SX-WINDOWをさらに便利に楽しく使うためのデスクアクセサリ集です。スクリーンセーバ、スクラップブック、スケジューラ、アドレス帳、電子手帳通信ツール、パズルなど、12種の豊富なアクセサリが収められています。 2MB、ver.3.0

●マルチタスク機能をはじめ通信環境がさらに充実

Communication SX-68K

CZ-272CWD 標準価格19,800円(税別)

通信環境をさらに高めたウィンドウ対応の通信ソフトです。マルチタスク機能により他のアプリケーションを実行中でも簡単に通信が可能。自動ログイン機能やプログラム機能、など豊富な機能をサポートしています。 2MB、ver.1.1

●FM音源サウンドエディタ

SOUND SX-68K

CZ-275MWD 標準価格15,800円(税別)

他のミュージックソフトで演奏中の音色を、簡単に作成、変更できるマルチタスク機能、またエディット、イメージ、ウェーブの3つの編集/確認モードを装備。作成中の音色も50曲の自動演奏でリアルタイムに確認、編集できます。 2MB、ver.1.1

●SX-WINDOW対応になってさらにパワーアップ

倉庫番リベンジ SX-68K ユーザー逆襲編

CZ-293A(130㎜FD)/CZ-293AWC(90㎜FD) 各標準価格6,800円(税別)

倉庫番10年にわたるユーザーの投稿など、新作306面が目白押し。まさに倉庫番の最強版がSX-WINDOW上で楽しめます。AI機能やエディット機能、キャラクタ変更機能も装備。半年で解けたらあなたは天才?です。 2MB、ver.1.1

PRO-68K シリーズ

●X68030/X68000対応

CZ-295LSD 標準価格44,800円(税別)

※メインメモリ2MB以上が必要です。

C compiler PRO-68KのX68030/X68000対応版。MPU68030、MC68882の命令セットに対応したアセンブラ、デバッガ、ソースコードデバッガを付属。またHuman68k ver.3.0、ASK68K ver.3.0にも対応。新たにGPIBライブラリ、MC68882対応フローートライブラリを付属しています。

※(2MB,ver.1.1)の表示は、メインメモリ2MB以上、SX-WINDOW ver.1.1以上が必要であることを示します。

※発売予定のソフトの画面は実物とは異なる場合があります。

●お問い合わせは…シャープ㈱電子機器事業本部(液映)システム機器推進プロジェクトチーム 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)へ シャープ株式会社

資料請求券 C2/ソフト Oh!X 3月

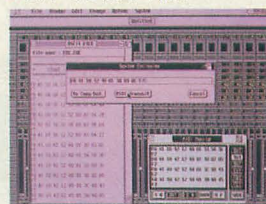
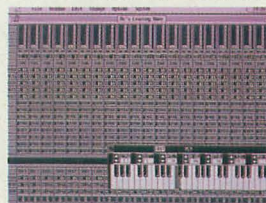
SOUND Canvas GS音源対応
MIDIマルチレコーダー

Mu-1 Musicstudio GS
[ミューワンジーエス]

Mu－1GSはローランド社SC－55mkⅡなどGS音源をフルに活用できるコントロール群と高度な音楽表現を可能にする新感覚エディットウインドウ搭載のMusicstudioプロフェッショナルバージョンです。

特長

1. 新感覚エディットウインドウ
   - エディタ感覚のプロフェッショナル仕様

2. 簡単エクスクルーシブ入力
   - チェックサム自動計算入力
   - 曲中でも使用可能！

3. ＧＳ音色コントロール機能
   コントロールコードのリアルタイムコントロールおよびステップ入力が可能
   - TVFカットオフフリケンシー、TVFレゾナンス、TVF&TVA・アタック、ディケイ、リリース・タイム
   - ドラムインストゥルメント・ピッチ、リバーブセンド コーラスセンド、パンポット、ボリューム

4. ＲＣＰコンバート機能追加
   - カモンミュージックRCM、ＳＴＥＤ2
5. 24トラック／リアルタイム録音／ステップ入力機能
6. Ｘ６８０３０(25MHz)／Human68K Ver.3.01対応
7. ＲＳ－２３２Ｃ／ＭＩＤＩ出力対応
   (注意：出力のみ対応、単独使用不可／要MIDIボード)
8. 内蔵ＦＭ／ＡＤＰＣＭ音源対応
9. 国本佳宏／ＧＳ対応デモ曲収録

データコンバート一覧表

読み込み（Load）		ファイル
Mu－1GS←	ミュージ郎／ミュージ郎Ⅱ MUSIC PRO-68K FM&MIDI MML MML スタンダードMIDIファイル フォーマット0/1 RCM／STED2	SNG MUS OPM MID RCP
書き込み（Save）		**ファイル**
Mu－1GS→	スタンダードMIDIファイル フォーマット1 MUSIC PRO-68K FM&MIDI	MID SCO

ハード構成 シャープ68000／030本体
ＭＩＤＩボード（シャープ社製CZ-6BM1またはシステムサコム社製SX-68M／SX68MⅡ）
ローランド社製GS対応音源SC-55、SC-55mkⅡ SC-300、SC-500など

Mu-1 GS 標準価格 ￥28,000（税別）

■本ソフト動作には、メインメモリ２MBが必要です。

スタンダードＭＩＤＩファイル
オリジナルアーティストシリーズ　各￥3,500

SCB-1001

SCB-1002

SCB-1003　SCB-1004

SCB-1001 duplicity／佐久間正英
SCB-1002 ブレインボックス美術館／国本佳宏
SCB-1003 PICES OF WORKⅡ／本多俊之
SCB-1004 HOPE／松居慶子

■推奨音源：Roland SC-55, SC-55mkⅡ, SC-88
SC-33, CM-500, CM-300

■「GS対応エクスクルーシブデータ」を使用しています。GM音源など推奨音源以外の機器を使用する場合、音量等のバランスが異なりますのでエディットしてお聴きください。

■SC-88対応「スタンダードＭＩＤＩファイル クラシックシリーズ」６タイトル発売予定

＜バージョンアップのお知らせ＞ Mu-1,Mu-1Super ユーザーの皆様へ
バージョンアップを行います。ご案内をお送り致しますので、未登録の方はユーザー登録ハガキをお送りください。

〒213 神奈川県川崎市高津区下作延１０４３
TEL ０４４－８５５－４３３５

新刊

X68k Programming Series #3

X680x0 TEX

吉野智興・川本琢二・山崎岳志・実森仁志●共著

●B5変形判・2冊組・ビニール箱入り●5"FD8枚組　定価9,800円

『Vol.1 User's Guide編』では、はじめてTeXを使う人のために簡単インストーラによるTeXの基本的な使い方の解説を、すでにTeXを使い込んでいる人のためにはカスタマイズのしかたや、数学記号などの表記に優れたAmSTeX、楽譜が書けるMUSIC-TeXなどのサンプルや、縦書きマクロ(アスキー、インプレス開発)などの周辺ツールの解説をしています。

また、『Vol.2 Reference編』では、TeX、METAFONT、fontman、preview、print、makefontなどの環境変数、オプションなどの解説をまとめてあります。

X68k Programming Series 追補版と改訂版 3冊同時発売【8月末予定】

X68k Programming Series ##

X680x0 Develop & libc II

吉野智興・中村祐一・石丸敏弘・今野幸義・村上敬一郎・大西恵司●共著

●B5変形判・5"FD2枚組●予価2,800円

「X68k Programming Series #1 X68000 Develop」収録のGCC、HAS、HLK、GDBと「X68k Programming Series #2 X680x0 libc」収録のライブラリをX68030でも動作するようバージョンアップした追補版です。

バージョンアップによって変更あるいは追加された機能と、約1年に渡るバグ報告を元に修正された機能について解説します。

付属FDには、最新のプログラムを収録しました。

X68k Programming Series #1

X680x0 Develop Manual Book

吉野智興・中村祐一・石丸敏弘・今野幸義●共著　●B5変形判・2冊組・箱入り●定価5,300円

X68k Programming Series #2

X680x0 libc Manual Book

村上敬一郎・大西恵司・萩野祐二●共著　●B5変形判・2冊組・箱入り●定価6,300円

それぞれ前作のマニュアル部分をまとめた改訂版です。

「X680x0 Develop & libc II」を発行するにあたり、変更・修正された機能についても解説しています。

近刊 X68000 マシン語プログラミング アルゴリズム編　著●村田敏幸

ソフトバンク株式会社出版事業部 〒103 東京都中央区日本橋浜町3-42-3 電話03(5642)8101

ゼッタイわかる！初心者のためのパソコン情報誌
Hello! PC
『Hello!PC』はあなたとパソコンをつなぐほっとラインです。
9月8日創刊
『Hello!PC』4つのポイント
『Hello!PC』はパソコン入門誌 の決定版！
従来のパソコン情報誌は、中級者向けのテクニックの提供が中心で、エントリーユーザーのニーズに応える情報誌はありませんでした。「初心者向け」「入門者向け」を謳った情報誌もありましたが、中身は大差ないものでした。『Hello!PC』は、エントリーユーザーのニーズに応えきる「パソコン入門誌」の決定版です。
『Hello!PC』はパソコンが本当に欲しくなる雑誌！
パソコンに関心はあるけれど、カタログを見ても意味がわからず、なかなか購入に踏みきれない……。これはエントリーユーザーが最初に抱える悩みです。『Hello!PC』は、必要最少限の専門用語しか用いません。エントリーユーザーが製品情報を読みこなし、かならず購入に踏みきれるように、やさしくガイドします。
『Hello!PC』はパソコンの魅力を新しく発見できる雑誌！
パソコンは、エントリーユーザーの思いになかなか応えてくれません。パソコン購入後に「こんなはずじゃなかった」と戸惑う人がたくさんいます。そんな時に必要なのは、パソコンの見方を少し変えてみることです。『Hello!PC』はエントリーユーザーがパソコンに新しい魅力、新しい夢を発見できるようにガイドします。
『Hello!PC』は理屈ではなく 肌でパソコンがわかる雑誌！
パソコンの世界にはいろいろなルールがあります。そうしたルールは、理屈で「わかろう、わかろう」としてもなかなかわからないものです。『Hello!PC』の誌面はビジュアル重視。イラストや写真を大きく扱い、平易なことばで読者の想像力に働きかけることで、パソコンのルールを肌で実感できるようにガイドします。
SOFT BANK
ソフトバンク株式会社/出版事業部
毎月8日発売　予価480円(税込)

GAMEBLAST
10月8日創刊
© 1993 Embodiment Films, AS Entertainment Planning & PIA

新製品紹介

MJ-700V2C

Taki Yasushi 瀧 康史

パーソナルユースでも普及しつつある高性能のカラープリンタ。しかし，X68000ではソフトウェアが対応していないのが悩みの種です。そこで，「ないものは作る」。このほど発売の新製品に対応のプログラムを瀧氏が開発することになりました。完成まではまだ少し時間がかかりますので，今月はまずその製品を紹介しましょう。

エプソンから，高画質カラー720dpiのプリンタが発売されました。メーカーからX68000対応と完全に謳われているわけではないのですが，高画質(高性能)・低価格といったコストパフォーマンスにつられて，X680x0への接続を狙い，思い切って購入してみました。読者のなかにもテレビコマーシャルを見て「いいかなぁ，でも，X68000との相性はどうかなぁ？」と考えていた人も多いことでしょう。

そういうわけで，私が「人柱」になってみました。

360dpiインクジェット(バブルジェットも含む)プリンタが出る以前のプリンタの選択は，カラー印刷ではインクジェット，文書印刷ではレーザープリンタという具合で，どちらも満足のいく仕上がりを期待するならば，2台のプリンタを購入する必要がありました。しかし，最近はインクジェットも360dpiで高画質なので，たいていの人は360dpiのインクジェットプリンタ1台で用が足りてしまいます。100枚以上の論文などのように多量の文章を印刷するならば，やはりレーザープリンタですけどね。

さて，このMJ-700V2Cは，プリンタの2つの用途である，カラー画像の印刷と文章の印刷を，美しくこなすことができます。解像度は標準モードで360dpi，インクジェットでA4サイズまでに対応と，ここまではごく一般の最近のインクジェットプリンタの性能です。お値段も99,800円と，このクラスではかなりお手頃です。これだけでもかなりお買い得ですが，このプリンタの最大の特徴は，720dpiモードという超高密度画質モードがあるということです。

720dpi時は特殊な専用紙を使わなければなりませんが，対応ソフトさえあれば，かなりの高画質で印刷できるはずです。

システムの対応

何はともあれ，まず使用してみました。

プリンタケーブルは別売りです。セットアップはMJ-700V2Cのマニュアルに書かれている順序で行います。これはホストマシンが何であろうと変わりません。プリンタにはディップスイッチらしきものがまったくないので，深いことは考えず接続します。

接続したらシステムを起動します。マニュアルにソフト側のプリンタ設定の優先順位が書かれているので，それを見ながら，SX-WINDOWのコントロールパネルで，ESC/P24-J84＊Cに合わせます。

そして，カットシートフィーダに紙を入れます。これで準備は完了。

これでSX-WINDOWからは，MJ-700V2Cは360dpiプリンタとして使用できます。私の記憶違いでなければ，ESC/P24-J84＊Cは24ドットカラーだったような気がしたのですが，なぜか360dpiプリンタで使えます。

シャーペンの印字サンプルをいくつか作ってみたので参考に見てください。ご覧のとおり，ごく普通に印刷できます。今月の連載「ローテク工作実験室」の回路も，Easy drawでMJ-700V2Cを用いて印刷したものです。

ただ，現時点のSX-WINDOWのカラー印刷については，文字印刷は問題ありませんが，カラーイラストを貼るとサンプルのようにかなり汚くなってしまいます。これはプリンタの性能の問題ではなく，SX-WINDOWのIVMのせいです。これさえ何とかなれば，360dpiでもSX-WINDOW上でかなり綺麗に印刷できるはずです。これはinc(IVMを作っているところ)に期待ということですか。

また，残念ながらX68000では720dpiモードが使えません。まあ，720dpiモードはスーパーファイン専用紙でしか使えないので，いずれにしても，現実には一般利用はほとんど360dpiモードになると思いますが。

MJ-700V2Cは，単に360dpiカラーと考えてみても，現在発売中の他機種に比べて安いので，総合的に考えるとなかなかよろしいんじゃないかと思います。SX-WINDOWで使う分にはまったく不自由しませんし。

TeXからの利用

X68000ユーザーの大半は，印刷にやっぱりシャーペンを利用するか，TeXシステムを利用することでしょう。

MJ-700V2Cは，基本的にエプソンのMJ-500V2がカラーになってスーパーファイ

MJ-700V2C　99,800円(税別)／エプソン販売　☎0424(99)7111

ンモードがついたというような感じです。それゆえ，MJ-500V2に対応しているソフトであれば，そのまま360dpiでモノクロ印刷することができます。

TeXは基本的にモノクロがベースですから，MJ-700V2Cでも普通の360dpiプリンタとまったく同じように印刷できます。ただ，90〜91ページで紹介されている「X68k Programming Series(#3) X680x0 TeX」に収録されているMJ500.cfgでは，印刷するとどうもおかしくなってしまったので，MJ-700V2C用の定義ファイルを作ってみました(リスト1)。

リスト1を入力して，環境変数として定義すれば，MJ-700V2Cで印刷が可能になるはずです。定義ファイル(デフォルトはprint.cfg) の構造がわからないままに，ちょこちょこいじっていたらなんとなくできてしまいました。知っている人から見れば，嘘っぱちがあるかもしれませんが，とりあえず，これで印刷できるので許してください。きっとそのうち，まともなものを誰かが作ってくれることでしょう。

360dpi，モノクロ印刷は，可もなく不可もなしといったところ。720dpiで印刷できるのならば，かなり綺麗になるのかもしれませんが，残念ながら，まだESC/Pコードがわかりません。

美しいカラー印刷を目指して

やっぱりせっかく720dpiのプリンタだから，それで印刷してみたい。というわけで，現在はプログラム制作のために資料を取り寄せています。本当は，ここでそのプログラムも掲載したかったのですが，製品マニュアルにはプリンタの制御方法は申し訳程度にしか書いていないうえ，このプリンタ自体，予約が多くてなかなか手に入らなかったため，今月号は製品紹介しかできませんでした。

それでも，360dpiでSX-WINDOWで美しいカラー印刷ができるかと期待していたのですが，満足のいくものではありませんでしたので，これはもう自分で作るしかありません。

とりあえず来月のおまけディスクには間に合わせたいところ。気合一発で作ろうとしているので，乞うご期待。

難点

このMJ-700V2Cの最大の難点は，ディップスイッチがないことです。MS-DOSとMS-WINDOWS，Macintosh対応の制御ソフトはありますが，SX-WINDOW用はありません。

どうやら，このプリンタは，プリンタ添付の「EPSON Remote!」というソフトで各種設定をしなくてはならないようで，そのソフトがないので，X680x0上から細かい設定ができないのです。

いまのところ，何も設定せずに使えますが，何かトラブルがあったときに，このソフトがないとどうなるかは謎です。

困ったらそのときに考えるということで，とりあえず「EPSON Remote!」はお預けでしょうか。

まとめ

現在，難しい設定など何もせずに，まったく普通のプリンタとして使用できています。その代わり，MJ-700V2C独自の機能は，まったく使えていないのが現状です。

ただ，どうやらDOSマシンで使っても，フルカラーでのA4印刷をするにはかなりたくさんのメモリが必要なようです。だから，X680x0版ではメモリが12Mバイトでもちゃんと印刷できるようにプログラムを組めば，もう完璧ってところでしょうね。

読者の皆さんには，とりあえず，360dpiでSX-WINDOW上で遊んでいてもらって，その先は，私がいま作っているプログラム待ちというところでしょうか。

カラー印刷(普通紙)

ターゲットコンピュータ
私がいま利用しているX
040turboを利用すれば，
z-16bit ステレオのデ
ータを適当な間隔で割り込
68000シリーズの10MHz
X68000-10MHzとい
く，快適度はMS-WINDO
感じている。X68000が1

コード印字(普通紙)

ターゲットコンピュー
私がいま利用している
ハードに近い)の040tur
匹敵するCPU処理能力を
込み，音声を出力しても
X68000の10MHzマシンで
X68000-10MHzという

シャーペンの印字サンプル(普通紙)

これは書体倶楽部、新明朝体です。
これは書体倶楽部、教科書体です。
これは書体倶楽部、ゴシック体（中）です。
これは書体倶楽部、ゴシック体（太）です。
これは書体倶楽部、毛筆体です。
これはZs'STAFFの、明朝体です。
これはZs'STAFFの、ゴシック体です。

This font is trad.
This font is Amadeus.
This font is Artist.
This font is baroque.
This font is Comp.
This font is House.
This font is Jack.
This font is Mabin.
This font is Pook.
[illegible]

やっぱり、使える半角がほしいよなぁ。

全部、48×48ドットフォントです。
因みにこれは内蔵フォント24ドットです。

各種のフォントを張り付けたもの(ファイン専用紙)

pcmplay.x は無料です。今後、
よって自由に使用し他人に譲ること
ブロード、商的目的で使用すること
ただし、pcmplay は全くのフリ
は著作権法で保護されていますし、

TeXの印字サンプル(普通紙)

リスト1　MJ-700V2Cの定義ファイル

```
-remark=  MJ-700  print.cfg
-dpi=360
-MSBisUpper
-pinBytes=6
-init=¥e@¥x18¥eA¥x08
-CRLF=¥r¥n
-FF=¥f
-graphic=¥e*¥x48%2i
-start=
-repeat=
-relative=
-xOffset=300
-yOffset=0
-width=2880
-height=3960
```

「GENIE」コンテスト応募作品

7月号で募集した「GENIE」コンテストにはたくさんの応募がありました。間違ってOh!X編集部に送った方，名前がどこにも書いてない方など，制作に熱中しすぎ(?)の方々もいらっしゃったようで……。ここでは応募作の一部をご紹介します。講評，データについては72～73ページをご覧ください。

小田島倫也さん(愛知県)

SF_001

SF_001

作者不明

PEM_G

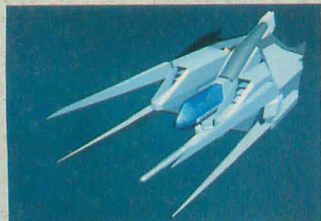
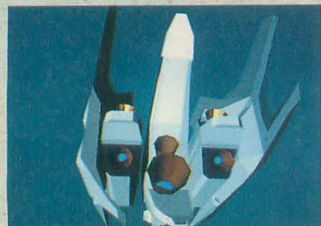
PEM_G

吉田昌之さん(茨城県)

KISBEY

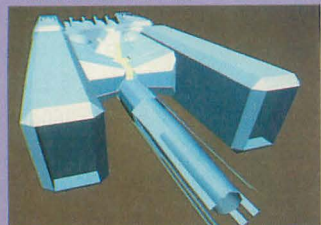
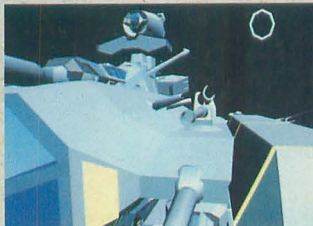
山本有樹さん(愛知県)

HAN

HAN

平山敏明さん(栃木県)

S_CRU

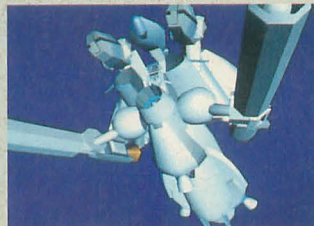
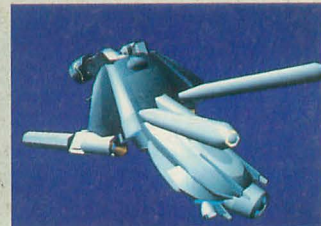
S_CRU

SS_P

小野寺英司さん(埼玉県)

INU6

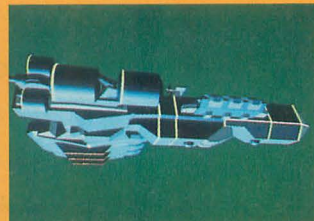
行本和弘さん(愛媛県)

OST

岡田行弘さん(岡山県)

REPORT

藤沢和行さん(大阪府)

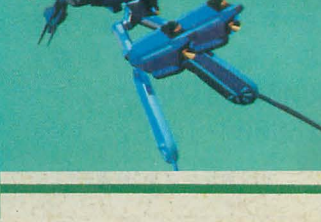
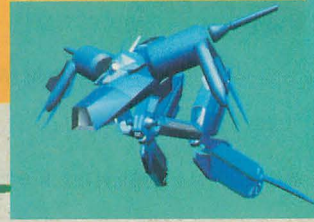
CRUSHER

CRUSHER

BASE

BIKE

TANK

ROBO

ROBOF

THE USER'S WORKS

●HAZARD2/Y2

今月は2作まとめて紹介しよう。シューティング風クイズゲームと本格派横スクロールシューティングゲームだ。かたや3700問の意欲作，もう一方はトレースモードつきの通好みの仕上がりとなっている。

シューティング風クイズだ。少しジャンルが片寄ってるが，問題数は豊富

●HAZARD2

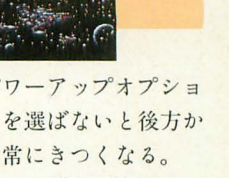
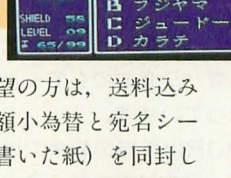
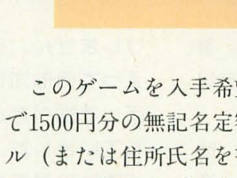
まずはTEAM NGのHAZARD2。一見，シューティングゲームのような雰囲気だが，内容はクイズゲームだ。

展開は横シューを模している。各エリアのザコキャラを蹴散らし，中ボスを倒し，ボスキャラをやっつければ1面クリアだ。クイズに正解すると1発ショットが撃てる。不正解ならダメージを受ける。見た目はグラディウスっぽいが1発死にではないのでご安心を。

解答ボタンを押す時間の早さに従ってタイムポイント（経験値のようなもの）が加算され，一定のタイムポイントになるとレベルアップしていく。ときどき敵がパワーアップカプセルを落としていくので，それを使ったパワーアップもできる。

ボスキャラ相手にはジャンル別の問題が出題される。ジャンルはゲーム関係，X68000関係，スーファミ関係などがあり，各ジャンルの下にはさらに対戦格闘，シューティング，必殺技などのサブジャンルがある。なお，一度選んだものはそのゲーム中は使えなくなる。

総問題数は3700問とかなりのものになっている。一般問題の難易度は低め，ジャンル別問題は……まあ，知らなくても25%は当たるのでなんとかなるか？（シールド関係はちょっと設定が甘すぎるような気がする）

全ステージクリアするとエンディングが始まり，クリア時間と正解率などを表示する。

このゲームを入手希望の方は，送料込みで1500円分の無記名定額小為替と宛名シール（または住所氏名を書いた紙）を同封して下記の住所まで連絡してほしい（ただし5インチ版のみ）。

〒569　大阪府高槻市萩谷月見台12-19
TEAM NG　橋爪　岳

●Y2

こちらは正真正銘のシューティングゲームである。ちょっとよくわからないタイトルだが，どうやらYAMIHIME2の略らしい。やはり意味がわからないが，ひょっとして主人公のことだろうか？

起動するとちょっとうっとうしいクレジットのあと，えんえん2分間デモが流れる。2人の女の子（つぐみちゃんとすずめちゃん）が戦闘機（ピンキー）に乗り込み……というと，なんか色モノっぽく聞こえるが，ゲーム内容は結構硬派。

背景グラフィックは地味めだが，マップ構成や仕掛けなどはなかなか見せるものがある。何箇所かに半透明を駆使した地形があるのだが，これは一長一短か。

設定によりR-TYPE風（？）とグラディウス風の2種類のオプション形態が選択できる。R-TYPE風のパワーアップオプション（デフォルトだが）を選ばないと後方から攻撃してくる敵が非常にきつくなる。

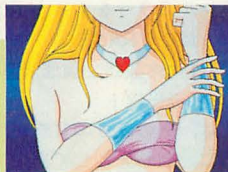
面が進むにつれ仕掛けはだんだん派手になるのだが，1面めが地味すぎるのが残念だ。単調な割にいやらしい攻撃が続くので後ろの面への意欲をそぐ。デモ画面との落差も大きいので，つかみが弱くなっている。

ゲームの難易度は中～やや高め。パターンゲームとはいえ，普通の人はEASYを選んだほうが身のためだろう。特にラスボスはちと手強い。

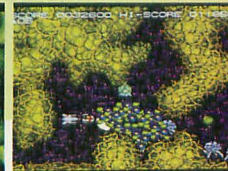
全6面，ディスク2枚組みで，ゲームには珍しいオンラインマニュアルも装備している。なによりいいのはトレースモードを装備していること。やはり，シューティングゲームにトレースモードがあると燃え方が違ってくる。

さて，このゲームはパソコンサークル西夢の会員が活動の一環として作り上げたものだ。こういうのはありそうでも，たいてい企画倒れになるものだが，よくまとめきったものだと思う。

このソフトを入手希望の方は，送料込み2000円分の無記名定額小為替と宛名シールを同封し，希望するソフト名とメディア(5インチ/3.5インチ）を明記して下記住所まで連絡してほしい。なお，3.5インチ版の発送は少し遅れるとのこと。

〒514　三重県津市渋見町630-3　西　敬史

アニメーションによるオープニングデモは本編とは関係ないかも。「撃ちまくり」というよりも「貫抜け/弾消し/避けまくり」といった要素が強い

[第40回]
影

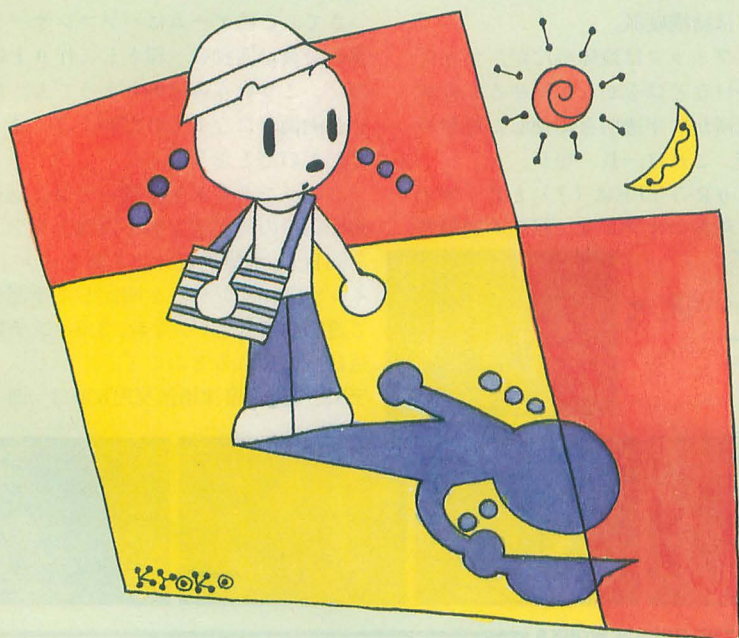
江口 響子

響子inCGわ〜るど

今年の夏は，昨年とうって変わっての猛暑。暑いですね〜近頃どうも夏バテ気味でして〜，というのが挨拶がわりになっています。

仕事場から最寄りのJR山手線の駅までは，徒歩12〜13分なのですが，日中30度を超えると，汗を流しながらせっせと歩くのを思い浮かべただけで，もううんざり。バス停にしたら2つほどの距離でも，つい冷房のきいたバスに乗ろうと思ってしまいます。バスが来るまでは暑さを少しでも和らげようと，待っている人のほとんどが建物の影に入ります。そのなかでホッとひといき。影のありがたさ。

こどものころは，夕暮れどきに友だちと追い駆けっこをして長く伸びた影を踏んだり，父に手で影絵を作ってもらったりして，ずいぶん遊びました。影はいつも身近にあり，けっしてなくなったりしません。そのため，とくに気にも止めず，かえって何も知りませんでした。そこで，図書館に行って調べたところ，おおよそこんなふうに記されていました。

「影は光が直進することによって生じる現象です。物体が光をさえぎると，光源と反対側に光線の届かない部分ができます。これが影です。光源がある大きさをもつとき，物体の後ろにできる影のなかで，光がまったく届かず，暗黒になる部分を，本影といいます。また，光が一部分到達して薄暗い影を作っている部分を半影といいます」

暗黒の影といっても，わたしたちが生活している空間では，周辺のさまざまな物体や空中のチリなどの乱反射によって，影の部分でも多少明るさをもっています。影に入っても，近くにいる人の顔が見分けられないほど暗いというわけではあり

今回のCGデータ

1920×1536ピクセル
1670万色フルカラーを4×5ポジで出力
使用ソフトはC-TRACE
総物体数54(物体数36，論理演算18)
点光源1，平行光線1
平行光線は，影の部分の質感がつぶれるのを防ぐために，補助光として当てています
「CGわ〜るど」のCGは，ピクセルの縦横比(アスペクト比)1：1でレンダリングしたものを，ポジ出力しています

ません。

でも，３次元CGの空間は，仮想の空間であり，こうした乱反射がないために，影の部分は光がまったく当たらず，文字通り暗黒につぶれてしまいます。そこで，環境光というものを設定して，光が回り込むようにし，不自然さを回避しています。

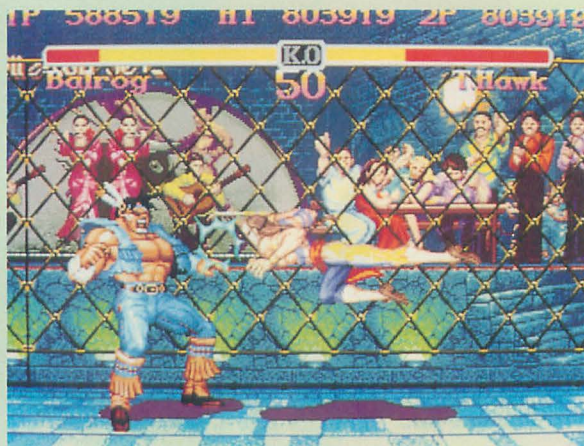
CGでは色を自由に変えることが可能です。環境光の色を青や赤にすると，影になるところが，その色になります。日常の空間では影の色は灰色ですから，仮想の空間で現実にはありえないような影の色を見ると，とても不思議な気分です。おとなになって，CGに出合い，また影で遊ぶようになるとは……。

影という言葉には，日本語ではロマンティックな感じのほかの意味もあります。

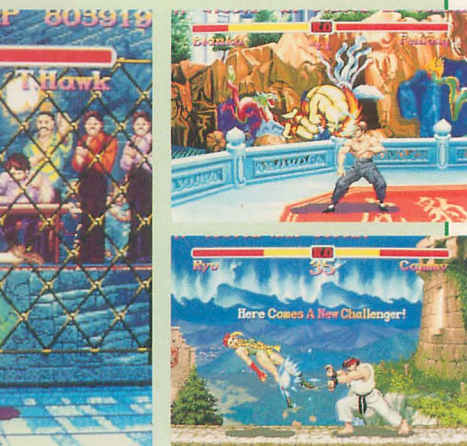
「心に思い浮かべた人の姿，おもかげや空想などによって心に思い描く実体のないものなど……」

こういうことは軽井沢や上高地あたりの避暑地の木陰でのんびりと考えたいものだ，と都心のマンションの小さな一室でつくづく思ったのでした。

SOFTWARE INFORMATION

格闘ゲームの新作が続きます。それにしてもどこからくるのか，この人気。やはり闘いは人間の本能なのでしょうか。さあさあキミも，パワーアップしていく新作を手に，たぎる血潮に火をつけろ！

スーパーストリートファイターII

日本ではこの夏アニメ映画公開，アメリカでは実写版映画の撮影も快調と，いまだ衰えぬストIIフィーバー。X68000には，昨年秋アーケードで話題を独占した「スーパーストII」が登場。最新作の「スーパーストIIX」でないのがファンには少し残念だが，開発に着手したと思われる昨年の冬頃にはまだスパIIXはなかったわけで，実は今回の移植プロジェクトはスパIIのアーケード登場の直後に始まったのかもしれない。

現時点での手元のサンプル版では，なかなかのデキ。前作で指摘された効果音の微妙な相違も，今回は徹底的に改善されている。新4キャラの技のキレも文句なし。移植度は限りなくアーケードそのまま。あとはBGMだな。で，メインメモリ要4M，奨励マシンクロック16MHz以上ということにも留意！　詳しくは来月。(善)

X68000用　5″2HD版　9,800円(税別)
カプコン

魔法大作戦

いっくぼ～んのヒット作「コットン」のEAビクターの新作は「魔法大作戦」。派手な画面の縦スクロールシューティングである。発売が予告されてから約半年，期待の声に応えてようやく新しい情報が寄せられた。

編集部に届けられたのはまだ開発途中バージョンなので，操作性やゲーム展開などはまったくわからないが，写真をご覧になれば，ゲームの雰囲気はつかめるだろう。

10月の発売に向けて，開発はいよいよ佳境にさしかかっているようだが，前作に続き，ユーザーの心をつかむ1作となりそうだ。

X68000用　5″2HD版　9,800円(税別)
EAビクター　☎03(5410)3111

画面は開発中のものです

クイーン・オブ・デュエリスト外伝＋α

発売は８月上旬なので，これを読んでいる皆さんのなかには，すでにお気に入りのキャラを使いこなしている人もいるかもしれない。以前にも述べたように，キャラクターごとにデザイナーが異なり，それぞれにまったく違う魅力があるので，やはり思い入れ度も大きくなるのではないだろうか。

格闘を楽しむほかに，ムフフなアレも期待する人には，パッケージ版の18禁バージョンもある。TAKERUブランドで9,800円(税別)だ。

X68000用 3.5/5"2HD版 5,800円(税込)
TAKERU ☎052(824)2493

プリンセスメーカー

他機種で大ヒットした子育てシミュレーションの元祖「プリンセスメーカー」。遅ればせながら，とうとうX68000版の発売が決定した。

ゲームはまず，自分の娘の名前をつけることから始まる。輝かしい未来への可能性を秘めた10歳の孤児。彼女は今日からキミの娘である。18歳になったときに彼女がどんな人生を勝ち取るか，すべてはキミの手にかかっている。

他機種版では30通りのエンディングすべてを見るのはなかなか大変，とのことだったが，X68000版ではどうなるだろうか。

来春の発売をお楽しみに。

X68000用 5"2HD版 14,800円(税別)
ニュー ☎0471(60)1131
画面はPC-9801版です

F-Card V5 for x68k

お手軽な表計算ソフト「F-Calc for x68k」のクレストが，カード型データベースソフト「F-Card」をバージョンアップする。会社名の曖昧検索やソート機能の追加などのほか，付属のフォントにより毛筆文字の印刷が可能になるとのことである。８月18日発売予定。

X68000用 3.5/5"2HD版 14,800円(税別)
TAKERU 14,800円(税込)
クレスト ☎03(3418)5993
TAKERU ☎052(824)2493

発売中のソフト

★餓狼伝説SPECIAL 魔法株式会社 7/28
X68000用 5"2HD版 9,800円(税別)
★レッスルエンジェルス３ TAKERU 7/31
X68000用 3.5/5"2HD版 5,800円(税込)
★クイーン・オブ・デュエリスト外伝＋α TAKERU 8/上
X68000用 3.5/5"2HD版 5,800円(税込)
★R.C.ロボット集＋α Vol.3 TAKERU・エレクトリックシープ 8/11
X68000用 3.5/5"2HD版 1,800円(税込)

新作情報

★スーパーストリートファイターII カプコン 9/30
X68000用 5"2HD版 9,800円(税別)
★魔法大作戦 EAビクター 10/未
X68000用 5"2HD版 9,800円(税込)
★X CASE Béシステム
X68000用 5"2HD版 19,800円(税込)
★Traüm 象スタジオ
X68000用 5"2HD版 価格未定
★鮫！ 鮫！ 鮫！ KANEKO
X68000用 5"2HD版 価格未定
★達人 KANEKO
X68000用 5"2HD版 価格未定
★エアバスター KANEKO
X68000用 5"2HD版 価格未定
★麻雀悟空・天竺への道 シャノアール
X68000用 5"2HD版 9,800円(税別)
★スタークルーザーII アルシスソフトウェア
X68000用 5"2HD版 価格未定
★地球防衛MIRACLE FORCE カスタム
X68000用 5"2HD版 価格未定
★XDTP SX-68K シャープ 6/未
X68000用 3.5/5"2HD版 価格未定
★プリンセスメーカー ニュー
X68000用 5"2HD版 14,800円(税別)
★VIEW POINT ネクサスインターラクト
X68000用 5"2HD版 価格未定
★F-Card V5 for x68k クレスト 8/18
X68000用 3.5/5"2HD版 14,800円(税別)
ブラザー工業(TAKERU)14,800円(税込)
★スターラスター 電波新聞社 8/26
X68000用 5"2HD版 5,900円(税別)

限界を追究のビッグタイトルなのだ

Komura Satoshi
古村　聡

あれからほぼ半年。魔法株式会社が満を持して送り出してきた「スペシャル」な最新作。心熱くした，あのキャラクターたちに再び会えるのだ。好みのキャラで頂点を目指してくれ。ところで，君はリョウ・サカザキに勝てるかな。

「大自然のおしおきよっ！」が先かと思ったんだが，でもよいっ！　NEO・GEOが爆発的に売れた原因といわれる「サムライスピリッツ」に並ぶビッグタイトル，餓狼伝説シリーズ最新作の「餓狼伝説SPECIAL」がX68000に移植されたんです。

この餓狼伝説SPECIALは，前2作で登場した主なキャラクターから選ばれた15人プラス隠れキャラとして「龍虎の拳」のリョウ・サカザキの計16人で，殴る蹴るの大壮絶バトル大会なんであります。とはいっても，第1作で出てきた，おいらの敬愛する超ボクサー，マイケルマックス様は出番なしなのですが……シクシク。いやあ，出ねーだろうなとは思ってましたけどね。だって技ないんだもん(苦笑)。

で，餓狼伝説といえばまっ先に思い出すのが「100メガショック！」という例の宣伝文句。餓狼伝説はNEO・GEOでも100メガを超えるROMカートリッジになっている超巨大作品なのですが，ROMのメガはメガビットだからバイトに直すと8分の1とはいえ，20メガバイト以上……。で，このX68000版，買ってきてぴっかぴかのパッケージを開けるとフロッピーディスクがいっぱい。そう，なんと9枚も入っているのです。

これを入れ替え差し替えするのか～？　つらいぞ～，と思うのは素人の浅はかさ。大丈夫，ちゃんとハードディスクやMOにもインストールできます。いとも簡単。ディスクを入れ，SHIFT キーと OPT.1 キーを押していると自動的にインストーラが起動します。でもって，指示どおりにディスクを入れ替えていけば……ほら，ハードディスクに餓狼伝説SPECIAL一丁上がり。で，あとは0ドライブに餓狼伝説のAディスクを入れて起動すると，ハードディスクにインストールされているかどうかを自動的に見分けて立ち上げてくれるんですよね。うーむ，偉い。

コンフィグレーションで好みのボタン配置に

リョウを含め16人からキャラを選べるぞ

ところで，NEO・GEOもゲームセンターのMVS版でもゲームに使うボタンは4つありました。このX68000版餓狼伝説SPECIALでは，餓狼伝説2の初期ロットに付属の，通称「魔法の4ボタンパッド」に対応しています。または電波新聞社製の6ボタンパッドアダプターを使えばセガ・メガドライブ用の6ボタンパッドも使えます。どちらも持ってないよー，という場合は普通のX68000用の2ボタンジョイスティックを使い，Aボタンを長く押すことで大パンチ，短く押すことで小パンチを出すことができます(きついなー)。

そうそう，ジョイスティックのボタン設定やキーボードを使う場合のキー割り当てはOPTIONメニューから設定できます。魔法のパッドではボタンが縦横2×2で配置されていますけど，デフォルトでは下2つがA，Cボタンで弱パンチ強パンチ，上2つがB，D，ボタンで弱キック強キックになっているんですよね。しかし，このスティックコンフィグレーションを使えば，これをゲームセンターのMVS筐体でよく設定されているように上2つをパンチ系，下2つをキック系に振り替えるなんてこともできちゃうってわけです。餓狼伝説2での問題点を解決しているんですね。偉い。

餓狼伝説SPECIALの特徴

でもって，餓狼伝説SPECIAL(以下SPECIAL)ではマックス君は出てこないわけですが(苦笑)，前作などではプレイヤーが使えなかったキャラクターも使うことができるようになりました。棒使いのビリー，ボクサーのアクセル，闘牛士のローレンス，不死身のギース，帝王クラウザー，それにVSモードでは龍虎の拳のリョウ・サカザキも使えます(リョウは隠しコマンドでストーリーモードでも使えるらしいけど……未確認)。X68000版の餓狼伝説2ではクラウザーたちも使えたけど，SPECIALのほうがバランスは抜群にいいみたいです。

では，SPECIALで変更された部分をキ

X68000だからタイガーキックもくっきりだ

X68000用　5"2HD版　9,800円(税別)
魔法株式会社　☎078(261)2790

ャラ別に見ていきましょう。まずはステージが新しくなったエラい人たちから。

[アクセル]

アクセルのステージは電線の張り巡らされたリングで闘う電撃デスマッチになりました。ライン移動しようとすると電線に触れて電撃バリバリスタンガン状態でダメージがでかいです。まるで，どこぞの過激なプロレス団体でんがな。でもって，アクセル君は技は当たるとでかいけど動きは遅いです。超必殺技のアクセルラッシュは速いけど，コマンドは複雑。むむ～，ちとつらいキャラクターかもしれないですね。

[ビリー]

餓狼伝説シリーズ全部に登場している洗濯屋ビリー君のステージは，どっかの城の時計塔の中。だと思います，たぶん。でもって，ここもアクセルのステージと同じくライン移動しようとすると奥の歯車に噛まれて血が噴き出すんです。ううっ，痛そう……。ちなみに3ラウンド目で引き分けると最終ラウンドではなぜか洗濯物の白いシーツがステージに出現します。まー，ステージの合間に洗濯機から出してきたんでしょーか？

あいかわらずおちゃめなビリー君。

[ダック]

吸い込む吸い込むなんでも吸い込め！のダックキングです。いや～，餓狼伝説1に比べてずいぶん使える人になったダック君であります。ひよこのPちゃんもらぶりぃ☆だし。巨大なディスプレイありライブステージありの巨大ディスコがダックのステージ。ハデハデ。しかし，同じハデ系なのに，チンさんが闘うと妙に場違いな気がするのはナゼ……。

[タン]

タン先生，っていうとなんかOh!Xスタッフのグラフィックプログラムの巨匠かと思ってしまうんですけど，ステージは水墨画のような岩と雲の世界がバックです。背中から巨大な影が現われる「撃放」や巨大化してぐるぐる回転する超必殺技「旋風剛拳」があるんだけど，いまイチ使えないような

嗚呼，兄弟対決！

この2人の挑発……情けなくて涙が出るっス

……。足元弱いし。

[ローレンス]

暗黒の闘牛士ローレンスは対戦だとブラッディスピンが強いです。ほとんどハメだと思うんですけど，あれ。で，バックにはなぜか猛牛が走り回っていて，奥に飛ばされると牛にはねられてしまいます。ライン移動不可な3ステージのなかではダメージ受けたときにいちばん痛そうです。

[クラウザー]

SPECIALではキャラクターの色がパレットで何パターンか変えられていて，青の舞とか結構人気があるみたいだけど，個人的に，いちばん美しいのはこのクラウザーの金色のパターンだと思ってます。うっつくしいんだわあ，これが。でもって，レッグトマホークは威力あるし，超必殺技のカイザーウエーブはコマンド簡単だし，闇の帝王クラウザー。強い！ 渋い！ イカスぞ，さすがはクラウザー様だ。

当て身投げが中段の攻撃にしか使えないのはつらいけど，やっぱ，上半身ハダカの男はいかすよな，本田様といい，ギースといい。マックスなんか，もうポイだ……。

で，ステージは城だか劇場だかで，後ろではオーケストラが演奏しています。よくわからないけどさすが帝王。でも，このステージだけはX68000版ではBGMが安っぽく聴こえてしまって悲しいっす。お気に入りなのに。とほほ。

[ギース]

ビルから落ちて死んだはずなのになぜか蘇ったという，不死身を誇る男，ギース様です。ちなみにストーリーモードでもほか

リョウは龍虎の拳の技もそのまま使えるぞ

舞ちゃんに燃え燃え～っ(ダックキング氏談)

金色パレットのクラウザー様は美しいのだ

のキャラでは，2Pが乱入してやられるとCOMがやられたのと同じ扱いになるのに，このギース様とクラウザー様だけはコンピュータが操るキャラを倒さないと倒したことにならず「不死身の私が相手をしよう」とまた対戦させられることになるんですよね～。さすが不死身の男。ステージはギース様の私邸(でしょう。たぶん)。甲冑が並んでるんだけど……なんちゅう趣味しとるんじゃ，このおっさんは。

さて，でもって，続いては前作と同じで，ステージも変わっていない主役級の8名様。

[テリー]

餓狼伝説2からほとんど変わっていない。でも，バーンナックルとパワーウエーブが気持ち速くなったので，使いやすいかな。

[アンディ]

小の斬影拳をはじめ，技がみんな遅くなって前作に比べるとかなり弱くなってしまった感じ。アンディ使い以外の人にはほっと安心かもしれないけど……。

[東]

東もアンディ同様弱くなってしまったキャラだと思います。避け攻撃が遅くなったし，タイガーキックを出したあとのスキがかなり大きくなった感じ。

[キム]

強い，強すぎるっす。イッキにパワーアップしてます，この人。飛翔脚が当たってなさそうなところでもちゃんと当たったことになるし，めくりでも入るし，頭踏まれて痛そうだし，あんまりこの人と対戦したくないっす。ちなみに超技・鳳凰脚，別名「キム乱舞」はコマンドも龍虎乱舞と同じ

これがタン先生の(使えねー)撃放だ

超技はなかなか出ない。跳びこんでほしい……

です。この人の技ってやられると痛そうなのばっかり……。あんた，友達なくすよ。

［ベア］

いままでベア＝遅い，というイメージだったけど，SPECIALではずいぶんスピードが上がったみたいです。特にドロップキックが効果的。ドロップキックはDボタンを8秒押しだけど，これは押し続けるほど威力が強くなります。また，ほかの技を出しても押し続けていればその間は威力が溜まるので，常にDボタンを押しっぱなしにしておくのがコツ。

［十兵衛］

全体に前作より使いやすくなったみたいです。タメ技系のタメ時間が減ったし。ただしダッシュ二本背負いが相手の起き上がりには使えなくなったのは痛いかな。新技の猫じゃらしは，対戦だと狙っているのがみえみえでつらいですけど。

［チン］

どうもまわりの人の話を聞いていると，弱くなったらしいです。実は前作でほとんど使ってないので私にはわからなかったりするんだけど。だって醜いんだもん。

［舞］

「シャチョウサン」「日本いちーぃ」の舞ちゃんはX68000では音声がクリアになったので，ちゃんと「花蝶扇」に聞こえるようになりました。ジャンプ中に下か斜め下に入れてDボタンで投げられる「夢桜」という技が増えました。でも，舞使いの人は技が変わろうがなんだろうが舞ちゃん使うから，どうでもいいのかも。燃え燃え☆　ってことで。

そうそう，これはSPECIALのキャラ全員にいえることなんですが，前作と違って，連続技が使えるようになりました。キャンセル技と通常技をうまく組み合わせて何段も入れてしまいましょう。

X68000はすごいぜ!

このゲーム，フロッピーだと入れ替えが大変だし，制約がいろいろあったりします。容量の問題で，キャラクターセレクト時にアニメーションしないとか……。だから，ハードディスクでプレイするのが絶対お得です。ない人はこの際だからハードディスクを買ってしまいましょう。最近は安いですから。もっとも，同じハードディスクでも，安くて遅いものと，高くて速いものにメモリをたっぷり載んでやるのとでは，全然違うんですけどね。お金かけるとNEO・GEO並みにゲーム展開が速くなるけど……安いハードディスクだとそれなりです。ああ，この世はやはりお金なのですね，同情するなら金をくれ，って世界であります。X68000もついにそういう時代になったんですね。それが悪いとは思いませんけど，自分が安いハードディスクだとちょっと悲しいっす。いや，単なるひがみなんですけどね。

で，この餓狼伝説SPECIAL。X68000版に移植される際に削られたり，変わった箇所もあります。たとえばアンディのステージは対岸をひよこがピヨピヨ歩いているはずなのに，いません。それから，ゲームのドキュメントにも書かれていますが，テリー，アンディ，舞の面では横スクロールが縮小されています。でも，グラフィッカーさんはそれを感じさせないほどがんばってくれています。最初は私，わかんなかったし。グラフィッカーさんは偉大です。うむ。

それになんといっても前作に比べて大幅に処理スピードがアップしてます。でもって，X68000のハードの制約をものともせず，よくNEO・GEO版そっくりに作ってあります。キャラの当たり判定もほぼそのままだし，たとえば，対COM戦で十兵衛だと相手が舞ならジャンプぴょんぴょんしながら小キックするだけで勝てるとか，そういう攻め方もそのまま使えます。遊びの部分でも，十兵衛のステージでラウンド4で衝立が「山田十兵衛」って文字になったり，アンディのステージでキムが空を飛んでいったりとかもそのまま入っているんですよねー。餓狼伝説2でも「よくやってるなー」と思ったんですが，SPECIALはもう超絶的にがんばってますね，魔法株式会社。「バーンナックル！」「龍炎舞！」と叫んだりのサンプリング音と，画像が鮮明な分，X68000版のほうがいいかもしれないです。定価も半分以下だし。

私，NEO・GEO版持ってます。でも，このX68000版は本当にすごいです。がんばって作ってるのがわかります。迷わず「買っちゃえ！」っておすすめしていいかと思っておりますです，はい。

とにかくエライ！

たいへんよくできました，まる

いま，NEO・GEO版を隣に並べて遊んでいるんですけど，なんか，ときどきNEO・GEO版より速いこともあるみたいです。うちのマシンは24MHz(そう，RED ZONE＋5インチドライブね)で速いマシンだから，ってことを差し引いても，やっぱりすごいことですよね，これって。Oh!Xでの写真撮影にはSAVESCを使ったんだけど，これ使っても画面が化けまくるくらいなんで，よっぽどすごいテクニックを使いまくってるに違いありません。

そうそう，ついでにいうとこのゲーム，15kHz，31kzHどちらのディスプレイモードにも対応しています。ただし，メニューから切り替えるので，DOS/V用のディスプレイを流用している人が遊ぶにはつらいですけどね。最初は必ず15kHzで起動してしまうため，メニュー自体が見えませんから。ディスプレイ切り替えショートカットキーは欲しかった気が……。

それから，保証はないけど，メモリの空き容量が2Mバイト以上あればハードディスクから起動もできます。Z-MUSICの音色ファイルをAディスクから1つコピーする必要があるみたいだったけど……。

X68000のハードの制約でサンプリング音は一度に1つしか再生してないけど，「餓狼伝説2」のように必殺技名が途中でほかのPCMに消されるということはないし，ユーザーの工夫次第では多重重ね合わせもできるらしい。いやいや，魔法株式会社，今回は本当に凝りましたね。

ついでに「龍虎の拳」からゲストでユリちゃんとか「サムライスピリッツ」のナコルルとか出してくれると完璧だったかもしれないけど，そこまで求めるのは酷ってもんでしょーね。ちなみにNEO・GEO版の「龍虎の拳2」では「餓狼伝説」からギースがゲスト出演してますね。関係ないけど。

総合評価	0 — 5 — 10
移植度	★★★★★★★★★
スピード	★★★★★★★
サウンド	★★★★★★★
グラフィック	★★★★★★★★
クラウザー様	★★★★★★★★★★

特集

SX-WINDOW 環境セットアップ

▼

ver.3.1でまた進化したSX-WINDOW。
システム自体には大きな変更はないものの，図を描くためのEasy draw，文章を書くためのシャーペンver. 3という2つの基本アイテムが揃ったことによりSX-WINDOWの環境は大きく進化した。パソコン上で行われる作業の大半がテキスト編集だということを考えればこれらが実用レベルになるということが非常に重要な意味を持つことがわかるだろう。
さらにコンソールのおかげでウィンドウ環境を抜けることなくさまざまな作業が行えるようになった。プログラム開発も開発ツールのおかげで表舞台に上がってきた。
そういったものをまとめて，「環境」が形作られるような状況にまでなってきているといえるだろう。
このような「環境」であることを目指して作成されたウィンドウシステムが，ようやく日常的な作業のほとんどをその上でこなせるようになってきているのだ。
ウィンドウ環境になって得るものと失うものはそれぞれ大きい。
失ったものを取り戻すための努力が続けられている。
これまでの環境とギャップがあるゆえに，現在の環境が持つアドバンテージを最大限に生かし，さらなるパーソナルコンピューティングの世界を求めていくことができる。
すべての要素をウィンドウ上で統合する日のために，現在の環境のポテンシャルと限界を知ることが必要であろう。完成された「環境」というのはまだどこにも実現されていないものなのだから。

▲

CONTENTS

より使いやすいテキスト環境のために

シャーペンをカスタマイズしよう

Nakano Shuichi 中野 修一

SX-WINDOWの環境を著しく向上させたシャーペン.X
そんなシャーペンをもっと使いやすく
あなたならではの設定を作ってみませんか

まったく個人的な話で申し訳ありませんが，SX-WINDOWの環境が実に快適になってきました。

理由はいくつかあります。ワークマシンをX68030に換えたこともそうですし，SX-WINDOW ver.3.1の力も大きく影響しています。そして，なにより1024×1024ドットフル画面での作業が可能になったことです（というより，これが可能になったからSX用のマシンを新調するにいたったわけですが）。

768×512ドットのSX-WINDOW表示画面は実画面モードでないとデスクトップが狭すぎますが，かといって実画面モードだとスクロールするのも困りものでした（しないよりはマシですが)。インタレースモードを使った1024×848ドットモードなどもありますが，ちらつきがひどくて普通のディスプレイでは使えませんでした。しかし，その広さと使い心地には感動的なものがありました。やはりX68000でもウィンドウアクセラレータの類が必要なのかなぁと思っていたのですが，それが，いまや，オシレータ交換は必要なものの，特別な周辺機器なしで1024×1024ドットの画面がくっきりと表示されるようになったのです。

本体は無改造でも，ディスプレイを換えれば（フル画面は難しいにしても）かなり広い画面を実現できるようなノウハウがたまってきましたし，標準ディスプレイでもそこそこ広い画面ができるようになってきました。ウィンドウ環境ではデスクトップの広さは，使い勝手の面で非常に大きな影響を及ぼします。

ハード的な環境が改善されたのと同等の比重を占めているのが，SX-WINDOW ver.3.1です。いわずと知れたことですが，シャーペンの力によるものです。

新しいシャーペン（ver.3.0）での日本語入力環境はX68000で実現された最高のものです。機能面ではワープロやDTPソフトとまではいきませんが，高速，柔軟，かつ馴染みやすいという特徴を持っています。

馴染みやすい理由として，基本操作がED.Xとほぼ同等なことが挙げられます。それでいてED.Xよりも機能は上です。個人的にはCTRL+Kでの行末までのデリートがちゃんとカットバッファに入るので非常に重宝しています。

もともと，SX-WINDOW ver.3.0とともにシャーペンver.1.0がサポートされたことでX68000の日本語入力環境はかなり改善されていました。ただ，インライン変換でないため，スクロールONだとHENWINのウィンドウがしょっちゅう画面外にいってしまったり，同時にサポートされたASK68K ver.3.0のキー操作が馴染みにくく，ASK68K ver.2.0のまま使う人が多かったり（ver.3.0でないとキー操作がかなり不便）と，いまひとつ本格的な用途には使えない状況にありました。

新しいシャーペンでは，ASK68K ver.2.0相当の操作で全角/半角変換や平仮名/片仮名変換ができるので，これまでASK68K ver.3.0を使わなかった人にもおすすめできます。

使いやすさの決め手は，ようやく実現された「清く正しい」インライン変換でしょう。

ASK68Kのカーソル位置変換モードや無変換モード以外のWP.Xの変換動作はインラインっぽく処理はしていても，清く正しいインライン変換とはいいがたいものがありました。入力場所で変換はできるものの，それまで入力されていたテキストが隠されたり，次候補などの選択では視点を移動させなければならなかったりと，不便なことが多かったのです。

エディタの操作性とワープロの日本語処理がうまく融合して，さらにウィンドウ環境のメリットがフルに生かせるのですからシャーペンは強力です。自然に手に馴染んで，自然に変換できる当たり前の環境があるってこんなにいいもんだったんですね。

さらにコンソールがサポートされたのでSX-WINDOWを抜ける必要がほとんどなくなりました。私がSX-WINDOWを抜けるのはグラフィックツールなど，特殊なツールを動かすときか，FC.Xなどでメモリが足りなくなったときだけですね。

こうして，SX-WINDOWは実用段階というよりすでに標準環境として使えるところまで成長したといっていいでしょう（あとはASK68Kがもう少しなんとかなれば……）。

シャーペンをカスタマイズするには

便利になったシャーペン。それをもっと便利で使いやすくする方法，それがカスタマイズです。エディタはユーザーの手の延長ですから，常に思いどおりに動いてくれなければ快適な生活は望めません。

シャーペンでは自由にキー定義その他が変更できます。標準状態で使っていて不満のない人は特に変更する必要はないのですが，独自の機能を加えたり，自分の好みにあわせて配置を変えたりすることで，さらに使い勝手がよくなります。

なにが便利で使いやすいか，使い慣れているかというのは人によって違います。ここで紹介するものは，あくまでも私がいろいろやってみて使いやすいと思うものを並べているだけです。反論がある人ももちろんいるでしょうし，もっといい方法論を持った人もいるでしょう。とりあえず参考程度に見ておいてください。

では，カスタマイズといっても具体的にどういったことができるのでしょうか。

まず，シャーペンにはシステムで想定しているタイプ.ENV，エディタ.ENV，シャーペン.ENV，コンソール.ENVといった4種類の設定例があります。そうです。コ

ンソールもシャーペンをカスタマイズして実現されたものです。

キーバインドを変えればほかのエディタのキー操作をシミュレートできますし，C言語プログラムのソースに特化したモードだとか，SX-BASICプログラムに特化したエディタに仕上げることもできるわけです。

文書を書くときとプログラムを書くときでは(基本操作は共通としても)，望まれる機能が変わってくることもあります。定義ファイルはいくつでも登録できるので，メニューやシンボルで起動する設定を変えたものを登録しておけば用途に応じた使い分けができるでしょう（ただしキーマップは共通になります)。

モード設定

最初にカスタマイズすべきものは各種モードのデフォルト設定です。すなわち，

フォントの種類
文字の大きさ
1行文字数
強制改行幅
編集モード
表示設定
印刷設定

などです。

これらはメニューで選択してモードを変えておき，環境メニューの「キーと環境を保存..」を選択実行することで設定されます。

フォントの種類や文字の大きさは特に注意しなくても前の設定が継承されるようになっていますのでいいのですが，そのほかのモード設定は結構重要になってきます。

必ずタブを表示するようにしたいとか，印刷時の倍率などは変わりませんから。

行数表示はページ内の行数がよいのか，全体量のわかる物理行表示がよいのか，字詰めによる折れ曲がりを気にしなくていい論理行表示がいいのかといった指定ができます。一般的にいって，プログラムは論理行表示，普通の文書は物理行表示が使いやすいでしょう。

私の設定では，フォントはROM24ドット，最小改行幅36，スクロール行数1，物理行表示，カラー表示（白地)，頁枠表示，シングルウィンドウモード，読み込み時環境を無視，あとはだいたいデフォルトのものを使っています。一部特殊なものもありますが，表示が遅かろうが，バックは白地，スクロール行数は1というのは強く主張しておきたいところです。

スクロール行数が大きく設定されているとスクロール速度は上がるのでなんとなく高速に見えますが，カーソル位置が上下に跳ねるので見ていて疲れてしまいます。速く進みたいならページスクロールがありますしね。ちょんちょんと押したときには1行ずつ，押し続けると加速する，といった仕様ならよかったのですが。

キー定義の種類

まず設定を行うにはキー定義の書き出しを行います。ここで書き出したファイルをいじって読み直すことによりキー定義などが変更できます。その状態で「キーと環境の保存..」を実行すれば設定状態が保存されます。

定義できる機能は操作キー別にいくつかの系列に分かれています。ざっと見てみましょう。

機能は大まかに，

CTRL系
ESC系
OPT.1系
ファンクションキー系

の4種類に分かれています。

前者3つはそれぞれ，コントロールファンクション，エスケープシーケンス，キーボードショートカットなどのように呼ばれることがあります。

ED.X系の設定では，コントロールファンクションにはテキスト編集機能が集められています。これは手をホームポジションに置いたまま1ストロークで操作できます。

それに対し，エスケープシーケンスは常に2ストロークで動作します。エディタにおける編集以外の機能が入っています。モードの設定などもここで行うことが多いですね。

OPT.1と併用される機能はキーボードショートカットと呼ばれます。アプリケーションの機能を直接呼び出すためのものですが，アプリケーションを問わずに共通の操作ができるようにある程度の規約が定められています。表1にSXガイドラインに示されるショートカットを示します。

しかしこれ以外にも暗黙のうちに標準的なショートカット割り当てというのは存在します。つまりMacintoshのショートカットキーです。表2にMacintoshユーザーインタフェイスガイドラインでシステム予約されているショートカットと多用される設定例をまとめておきます。もともとがMacintoshの文化ですので，こういうものは揃えておいたほうがよいのかもしれません。

F1～F10のデファイナブルファンクションキーやDEL, HOMEなどの単一機能キーはまとめてファンクションキーと呼ばれます。デファイナブルファンクションキーは割り当て自由，機能キーはまあ普通の割り当てにしておくのが無難でしょう。

実際にはこれらのものに加え，3系統の

常用フォントの指定

文書を作るときには画面上では16×16ドットフォントが使いやすいのですが，プリントアウトを考えるとそうもいきません。

もちろん印刷時にいちいち書式を変えてやればいいのですが，それも面倒です。

現在私は表示/印刷ともに24ドットROMフォントを常用しています。プリンタは360dpiなので印刷時の倍率は200%です。

RAMディスクにフォントを置き，040turboを使えばほとんどROMフォント感覚でアウトラインフォントが使えるのですが，肝心のフォントに気にいったものがなく，結局ROMに落ち着きました（スムージングなどは品質を落とすだけなので指定しない)。これだとプリンタの性能を100%引き出すことはできないのですが，しかたありません。個人的にROMフォントのフォントデザインはかなり高く評価しています。

イメージ印字でなくてもよい場合は田村氏によるsetkindl.exを使うのもよいでしょう。

表1 SXの予約ショートカット

OPT.1+X	カット
OPT.1+C	コピー
OPT.1+V	ペースト
OPT.1+A	全選択
OPT.1+Z	取り消し

表2 Macintoshのショートカット

●システム予約されているもの

Com-N	New	(FileMenu)
Com-O	Open..	(FileMenu)
Com-W	Close	(FileMenu)
Com-S	Save	(FileMenu)
Com-P	Print..	(FileMenu)
Com-Q	Quit	(FileMenu)
Com-Z	Undo	(EditMenu)
Com-X	Cut	(EditMenu)
Com-C	Copy	(EditMenu)
Com-V	Paste	(EditMenu)
Com-A	Select All	(EditMenu)
Com-.	Terminate an operation	(EditMenu)

その他，スペースキーと左右カーソルも予約

●多用されるもの

Com-F	Find	(FileMenu)
Com-G	Find Again	(FileMenu)
Com-T	Plain Text	(StyleMenu)
Com-B	Bold	(StyleMenu)
Com-I	Italic	(StyleMenu)
Com-U	Underline	(StyleMenu)

ユーザーキーと，2系統のプレフィックスキーの動作が定義できます。

ユーザーキーとはコントロールキーを拡張したようなもので，シャーペンの内部ファンクションや外部コマンドを組み合わせて作成した機能を任意のシフトキーを使って（コントロールキーのように）実行できるようにするためのものです。オプションキーやコントロールキーの拡張ですね。

欠点はというと，OPT.2以外のキーはすべて変換キーになっていますから，これらを割り当てると日本語変換モード時に範囲指定して機能させることができないということです（先に変換モードになってしまう）。

押す必要がないだけで，確かにWP.Xなどでもそういう動作だったのですが，インライン変換のような凝ったことをやるくらいなら，これくらいなんとかなったんではないかという気が……。

プレフィックスというのは，ESCキーの動作がそれにあたります。emacsを使っている人にはもうお馴染みでしょう。メタキーの使い方と同じです。おそらくemacsのキーバインドを実現するために追加された機能なのでしょう。

そのほか「どうしてもエディタの終了は，CTRL-K+CTRL-Dじゃなきゃやだ」という人ならWordstar系のキーバインドにすることもできるはずです（たぶん……。詳しくは後述）。

変更してみる

では実際にキー定義を変更してみましょう。

キー設定は，標準状態では「ほぼ」ED.X互換となりますが，いくつかの点で違いが見られたり，旧版のシャーペンから変更されたものがあります。手始めにそれらの点を修正してみましょう。

●マクロの登録

エディタに慣れてくるにつれ多用されるコマンドですが，ED.Xでは，

ESC @

だったものが，なぜかシャーペンでは，

CTRL+O

となっています。

代わってESC @に登録されているものは広域の置換コマンドです。

シャーペンの場合，たくさんのテキストを開くことがありますが，どれも同じシャーペン.Xによって起動されているので，広域指定はそのすべてにかかってしまいます。

開かれているのは必ずしも同じ系統のテキストファイルだけではないということです。ソースプログラム群を2系統開いておいたり，ドキュメント類があってもどれとどれにだけ検索/置換を行うという指定ができません。これでは困ることもあります。

特に私の場合，常時シャーペンで10～30タスク以上のテキストを開いていますから，広域の置換（検索はともかく）はかなり危険な処理になります。

よって，広域置換は使わないか，少なくとも使用頻度は低いので，ESC @にはマクロ登録を優先しました。

指定は，

#15　　＊ ESC ＋ @

となります。「＊」は注釈の指定ですが，定義ファイルのこれと同じ注釈の部分を書き換えてみてください（以下，同じ手順）。

●ファイル切り換えの変更

以前のバージョンではHOMEとCLRキーで編集ファイルを前後に移動できたのですが，ウィンドウ分割が拡張されたためか，これらのキーの分担が変わっています。

ファンクションキーの部分を，

表3　デフォルトのコントロールキー設定

コントロールキー		
^@	#32	なにもしない
^A	#1	カーソルを1ワード左に移動
^B	#2	カーソルを左端（または右端）に移動
^C	#3	画面をロールアップ
^D	#4	カーソルを1文字右に移動
^E	#5	カーソルを1文字上に移動
^F	#6	カーソルを1ワード右に移動
^G	#7	1文字削除
^H	#8	バックスペース
^I	M1,'tab',$0D	水平タブ
^J	#10	マークの登録
^K	#11	カーソル位置から行末までを削除⇒削除バッファへ
^L	#12	削除バッファの内容をカーソル位置に複写
^M	#13	改行と行分割
^N	#14	カーソルの上に1行挿入
^O	#113	キーボードマクロの定義開始，および定義終了（終了後にキーバインドする）
^P	#16	カーソルを右端に移動
^Q	#17	カーソルを左端に移動
^R	#18	画面をロールダウン
^S	#19	カーソルを1文字左に移動
^T	#20	1ワード削除
^U	#21	行頭からカーソル直前までを削除⇒削除バッファへ
^V	#22	コントロールコードの入力
^W	#23	画面を設定したスクロール行分ロールダウン
^X	#24	カーソルを1文字下に移動
^Y	#25	カーソル位置の1ラインを削除⇒削除バッファへ
^Z	#26	画面を設定したスクロール行分ロールアップ
^[	#27	ＥＳＣコマンド
^¥	M1,'sea -C -B',$0D	カレントワードの後方検索
^]	M1,'case',$0D	現在のモードで大文字／小文字変換
^^	M1,'sea -C',$0D	カレントワードの前方検索
^_	#175	キーボードマクロの中断（確認あり）
SHIFT + ^@	#32	なにもしない
SHIFT + ^A	#161	現在の選択位置を1ワード左に移動
SHIFT + ^B	#162	現在の選択位置を左端（または右端）に移動
SHIFT + ^C	#163	現在の選択位置を1画面下に移動
SHIFT + ^D	#164	現在の選択位置を1文字右に移動
SHIFT + ^E	#165	現在の選択位置を1文字上に移動
SHIFT + ^F	#166	現在の選択位置を1ワード右に移動
SHIFT + ^G	#7	1文字削除
SHIFT + ^H	#8	バックスペース
SHIFT + ^I	#9	水平タブ
SHIFT + ^J	#215,#70,'-R -W',M1,'adrect-W50,50,606,430',$0D,' ',M1,'getpath -P0 -E -X1',$0D,'.HLP',#216,$0D	ヘルプファイルを開く
SHIFT + ^K	#11	カーソル位置から行末までを削除⇒削除バッファへ
SHIFT + ^L	#12	削除バッファの内容をカーソル位置に複写
SHIFT + ^M	#13	改行と行分割
SHIFT + ^N	#14	カーソルの上に1行挿入
SHIFT + ^O	#15	キーボードマクロの定義開始，および定義終了
SHIFT + ^P	#176	現在の選択位置を右端に移動
SHIFT + ^Q	#177	現在の選択位置を左端に移動
SHIFT + ^R	#178	現在の選択位置を1画面上に移動
SHIFT + ^S	#179	現在の選択位置を1文字左に移動
SHIFT + ^T	#20	1ワード削除
SHIFT + ^U	#21	行頭からカーソル直前までを削除⇒削除バッファへ
SHIFT + ^V	#22	コントロールコードの入力
SHIFT + ^W	#183	画面を設定したスクロール行分ロールダウンし選択位置を変更する
SHIFT + ^X	#184	現在の選択位置を1文字下に移動
SHIFT + ^Y	#25	カーソル位置の1ラインを削除⇒削除バッファへ転
SHIFT + ^Z	#186	画面を設定したスクロール行分ロールアップし選択位置を変更する
SHIFT + ^[	#127	ＥＳＣコマンド
SHIFT + ^¥	M1,'sea -C -B',$0D	カレントワードの後方検索
SHIFT + ^]	M1,'case -S-1',$0D	現在と反対のモードで大文字／小文字変換
SHIFT + ^^	M1,'sea -C',$0D	カレントワードの前方検索
SHIFT + ^_	#175	キーボードマクロの中断（確認あり）

表4　デフォルトのエスケープシーケンス設定

エスケープシーケンス		
ESC + ^A	#167	編集ウィンドウの切り替え（昇順）
ESC + ^B	M1,'sea -C -G2',$0D	カレントワードの広域検索
ESC + ^C	#221	外部コマンドの起動
ESC + ^D	#168	編集ウィンドウの切り替え（降順）
ESC + ^E	未定義	
ESC + ^F	#102	新規編集ウインドウのオープン
ESC + ^G	未定義	
ESC + ^H	#206	環境の保存
ESC + ^I	M1,'sea -G2 -L',$0D	前回検索した文字で広域検索
ESC + ^J	M1,'rep -L',$0D	前回検索した文字で前方置換（確認あり）

#101　　＊ CLR
#99　　 ＊ HOME

に，シフト＋ファンクションキーの部分を，

#98　　 ＊ CLR
#100　　＊ HOME

にすれば，HOME，CLRで以前と同じ動作，シフトを併用すると表示されているファイル間での移動になる，はずです（うまく動作しない気もするが）。

シャーペンver.3.0からウィンドウ内部を縦横に分割できるようになりましたが，非アクティブなウィンドウへの操作もユーザーキーの3を併用することで可能になっています。2分割して両方のウィンドウを同時にスクロールするといったこともできます。

なお，最大分割数はまだ未確認です（少なくとも24個までは分割できる）。

ショートカットキーの設定

シャーペンにはさまざまな機能キーの系列がありますが，OPT.1の設定だけはメニュー中でショートカットキーの割り当てが表示できます。キーボードを主体で使うテキストエディタでショートカットもなにもないような気がしますが，単にメニュー中にショートカットのマークとキー割り当てが表示されるだけでも多少は使いやすくなります。

シャーペンではファイル関係やカット＆ペースト関係のコマンドはED.X準拠のESC系列とOPT.1系列のキーボードショートカットの2系統があります。

おそらくED.X系のコマンドとMacintosh系のキーボードショートカットを交ぜただけだと思いますが，そのため一部の機能は両方で定義されています。

どちらかというとストローク数の分だけショートカットのほうが使いやすいので，よく使うものをショートカットにも割り当ててみましょう（もともとMacintosh系の文化なので変に拡張しないほうがいいものなのかもしれませんが……）。

メニューに設定されていて多用される（少なくとも私は多用する）機能として「すべて保存」があります。多用しなくてすむならそのほうがいいのですが，何度となく痛い目を見ると，いやでも頻繁にファイルセーブするようになります。現在編集中のファイルをセーブするだけならショートカットキーとして割り当てられているのですが，それだけでは裏に隠れたファイルを失うことがあるので「すべて保存」もショー

ESC + ^K	未定義	
ESC + ^L	M1,'rep -L -B',$0D	前回検索した文字で後方置換（確認あり）
ESC + ^M	#109	改行文字の表示／非表示
ESC + ^N	M1,'sea -L',$0D	前回検索した文字で前方検索
ESC + ^O	未定義	
ESC + ^P	未定義	
ESC + ^Q	未定義	
ESC + ^R	M1,'rep -L -Q',$0D	前回検索した文字で前方置換（確認なし）
ESC + ^S	M1,'sea -L -B',$0D	前回検索した文字で後方検索
ESC + ^T	#78	環境ファイル名変更
ESC + ^U	M1,'rep -L -Q -B',$0D	前回検索した文字で後方置換（確認なし）
ESC + ^V	#83	バージョン表示
ESC + ^W	未定義	
ESC + ^X	未定義	
ESC + ^Y	#74	環境ファイルの読み込み
ESC + ^Z	未定義	
ESC + ^[	未定義	
ESC + ^¥	未定義	
ESC + ^]	未定義	
ESC + ^^	M1,'rep -L -H',$0D	前回検索した文字で前方置換（表示なし）
ESC + ^_	M1,'rep -L -H -B',$0D	前回検索した文字で後方置換（表示なし）
ESC +	#97	すべてのテキストを保存して編集は継続
ESC + !	#181	1ライン半角文字数入力
ESC + "	#182	水平タブ文字数入力
ESC + #	#35	改行幅入力
ESC + $	#36	スクロール行数入力
ESC + %	未定義	
ESC + &	未定義	
ESC + '	#39	テキスト終了記号の表示
ESC + (	#44	タブ文字の表示
ESC +)	#41	（カット）指定した範囲を削除⇒クリップボードへ
ESC + *	#42	（コピー）指定した範囲　⇒ クリップボードへ
ESC + +	#60	指定行数前方へジャンプ
ESC + ,	#215	表示オフ
ESC + -	#62	指定行数後方へジャンプ
ESC + .	未定義	
ESC + /	#47	カーソルをホーム位置に移動
ESC + 0	#48,'0'	数値付コマンド入力
ESC + 1	#48,'1'	数値付コマンド入力
ESC + 2	#48,'2'	数値付コマンド入力
ESC + 3	#48,'3'	数値付コマンド入力
ESC + 4	#48,'4'	数値付コマンド入力
ESC + 5	#48,'5'	数値付コマンド入力
ESC + 6	#48,'6'	数値付コマンド入力
ESC + 7	#48,'7'	数値付コマンド入力
ESC + 8	#48,'8'	数値付コマンド入力
ESC + 9	#48,'9'	数値付コマンド入力
ESC + :	M1,'key -C',$0D	キーボードマクロの編集
ESC + ;	#59	キーボードマクロをn回実行する
ESC + ;	#59	キーボードマクロをn回実行する
ESC + <	#116,#107,'86,22,556,162',$0D	小ウインドウ化 {96,16,556,162}
ESC + =	#61	画面大ウインドウ化
ESC + >	#116,#107,'86,22,680,490',$0D	大ウインドウ化 {86,22,680,490}
ESC + ?	#63	新規保存
ESC + @	M1,'rep -G2',$0D	広域置換
ESC + A	#99	編集テキストの切り替え（昇順）
ESC + B	#66	カーソルをファイルの先頭に移動
ESC + C	#67	コマンドの実行
ESC + D	#101	編集テキストの切り替え（降順）
ESC + E	#69	すべてのテキストを保存して編集を終了
ESC + F	#70,'-Gｴﾃﾞｨﾀ.ENV '	新しいファイルの編集
ESC + G	#71	クリップボードの内容をカーソル位置にn回複写
ESC + H	#72	編集中のテキストを保存して編集は継続
ESC + I	M1,'sea -G2',$0D	広域検索
ESC + J	M1,'rep',$0D	前方置換（確認あり）
ESC + K	#75	編集中のテキストを保存せず編集を終了
ESC + L	M1,'rep -B',$0D	後方置換（確認あり）
ESC + M	#77	マーク位置へのジャンプ
ESC + N	M1,'sea',$0D	前方検索
ESC + O	#79	テキストの編集を最初からやり直す
ESC + P	#80	カーソル位置からn行削除削⇒クリップボードへ
ESC + Q	#82	すべてのテキストを保存せず編集を終了
ESC + R	M1,'rep -Q',$0D	前方置換（確認なし）
ESC + S	M1,'sea -B',$0D	後方検索
ESC + T	#84	編集中のテキストのファイル名を変更
ESC + U	M1,'rep -Q -B',$0D	後方置換（確認なし）
ESC + V	M1,'tagjump -Gｴﾃﾞｨﾀ.ENV',$0D	タグジャンプ
ESC + W	#87	指定した範囲をファイルに書き出す
ESC + X	#88	編集中のテキストを保存して編集を終了
ESC + Y	#89	ファイルを読み込み，カーソル位置に挿入
ESC + Z	#90	カーソルをファイル最終行に移動
ESC + [	M1,'rep -C',$0D	カレントワードの前方置換
ESC + ¥	M1,'rep -C -B',$0D	カレントワードの後方置換
ESC +]	M1,'case -M-1',$0D	大文字／小文字変換モードを逆にする
ESC + ^	M1,'rep -H',$0D	前方置換（表示なし）
ESC + _	M1,'rep -H -B',$0D	後方置換（表示なし）
ESC + `	未定義	
ESC + a	#98	編集テキストの切り替え（昇順）
ESC + b	#66	カーソルをファイルの先頭に移動
ESC + c	#67	コマンドの実行
ESC + d	#100	編集テキストの切り替え（降順）
ESC + e	#69	すべてのテキストを保存して編集を終了
ESC + f	#70	新しいファイルの編集
ESC + g	#71	クリップボードの内容をカーソル位置にn回複写
ESC + h	#72	編集中のテキストを保存して編集は継続
ESC + i	M1,'sea -G2',$0D	広域検索
ESC + j	M1,'rep',$0D	前方置換
ESC + k	#75	編集中のテキストを保存せず編集を終了
ESC + l	M1,'rep -B',$0D	後方置換
ESC + m	#77	マーク位置へのジャンプ
ESC + n	M1,'sea',$0D	前方検索
ESC + o	#79	テキストの編集を最初からやり直す
ESC + p	#80	カーソル位置からn行削除削 ⇒クリップボードへ
ESC + q	#81	すべてのテキストを保存せず編集を終了
ESC + r	M1,'rep -Q',$0D	前方置換（確認なし）
ESC + s	M1,'sea -B',$0D	後方検索
ESC + t	#84	編集中のテキストのファイル名を変更
ESC + u	M1,'rep -Q -B',$0D	後方置換（確認なし）
ESC + v	M1,'tagjump -Gｴﾃﾞｨﾀ.ENV',$0D	タグジャンプ
ESC + w	#87	指定した範囲をファイルに書き出す
ESC + x	#88	編集中のテキストを保存して編集を終了
ESC + y	#89	ファイルを読み込み，カーソル位置に挿入
ESC + z	#90	カーソルをファイル最終行に移動
ESC + {	未定義	
ESC + ¦	未定義	
ESC + }	M1,'print2 -P-1',$0D	論理行／物理行の表示切り替え
ESC + ~	未定義	
ESC +	未定義	

トカットにしてみましょう（単に気分の問題ですが）。

ちなみに「すべて保存」はデフォルト設定ではESC系の割り当てで，ESC （スペース）となっています。

まず，OPT.1キーへの機能割り当ての空いた位置を探さなければなりません。デフォルト状態では同等なキー指定となるOPT.1+ （スペース）も空いているのですが，これだとメニューに表示されたときにわけがわからなくなりますので，ここではED.Xでのセーブ終了と同じ系列のEを使ってみました。書式は，

#97　　　＊ OPT.1 ＋ E

となります。

メニューをいじってみる

続いてメニューをエディットします。

まず，ウィンドウ幅を小さくしたときにメニューアイコンが見えなくなりますので少し整理しましょう。

まず不要な項目があれば削ります。

検索/置換機能がメニューに割り当てられているのは初心者には必要な配慮ですが，長年ED.Xを使ってきた人には必要ないものでしょう。私はこのメニュー項目をまるごと削りました。

さらに，網掛けはほとんど使うことはないと思われるので文字装飾関係のメニューに移して，「網掛け」の名前でサブメニュー化しました。実に使いにくそうな位置ですが，使うことはまずないでしょうからまったく問題ありません（きっぱり）。

フォントの種類選択と大きさ選択はほぼ同系統の処理とみなせますので，まとめてしまいます。2つのメニューをあわせてもそれほど長くなりません。

以前のバージョンを使っていたときは罫線も使わないのでまるごと削っていたのですが，ver.3.0では罫線が「使える」仕様になっているので残してあります。

これでメニューアイコンが7つに整理されました。

次にメニューの中身を検討します。メニューが不必要に長くなると操作性は低下します。

ファイルメニューにある「新規」というのは「開く..」で代用できますので削りました（「開く..」のほうが使いやすい）。

さらにメニューの表示文字列を検討します。毎度お馴染みの階層メニューの開き方なのですが，メニューの横幅が長いとサブメニューまでのマウス移動距離が大きくな

表5　デフォルトのオプションキー設定

オプションキー		
OPT.1 +	未定義	
OPT.1 + !	未定義	
OPT.1 + "	未定義	
OPT.1 + #	未定義	
OPT.1 + $	未定義	
OPT.1 + %	未定義	
OPT.1 + &	未定義	
OPT.1 + '	未定義	
OPT.1 + (	未定義	
OPT.1 +)	未定義	
OPT.1 + *	未定義	
OPT.1 + +	未定義	
OPT.1 + ,	#59	キーボードマクロをn回実行する
OPT.1 + -	#175	キーボードマクロの中断（確認あり）
OPT.1 + .	未定義	
OPT.1 + /	未定義	
OPT.1 + 0	M1,'cont',$0D	最新の編集位置へのジャンプ
OPT.1 + 1	#221	外部コマンドの起動
OPT.1 + 2	#67	コマンドの実行
OPT.1 + 3	未定義	
OPT.1 + 4	未定義	
OPT.1 + 5	未定義	
OPT.1 + 6	未定義	
OPT.1 + 7	未定義	
OPT.1 + 8	未定義	
OPT.1 + 9	未定義	
OPT.1 + :	未定義	
OPT.1 + ;	未定義	
OPT.1 + <	未定義	
OPT.1 + =	未定義	
OPT.1 + >	未定義	
OPT.1 + ?	未定義	
OPT.1 + @	未定義	
OPT.1 + A	#107	全選択
OPT.1 + B	#160	箱挿入
OPT.1 + C	#42	（コピー）指定した範囲　⇒ クリップボードへ
OPT.1 + D	M1,'split -Y',$0D	カレントテキストの分割
OPT.1 + E	未定義	
OPT.1 + F	未定義	
OPT.1 + G	未定義	
OPT.1 + H	未定義	
OPT.1 + I	未定義	
OPT.1 + J	未定義	
OPT.1 + K	未定義	
OPT.1 + L	#22,#12	強制改ページ
OPT.1 + M	未定義	
OPT.1 + N	#102	新規編集ウインドウのオープン
OPT.1 + O	#70	新しいファイルの編集
OPT.1 + P	未定義	
OPT.1 + Q	#81	すべてのテキストを保存せず編集を終了
OPT.1 + R	未定義	
OPT.1 + S	#72	編集中のテキストを保存して編集は継続
OPT.1 + T	未定義	
OPT.1 + U	未定義	
OPT.1 + V	#45	(ペースト)クリップボードの内容をカーソル位置に
OPT.1 + W	#75	編集中のテキストを保存せず編集を終了
OPT.1 + X	#41	(カット)指定した範囲を削除⇒クリップボードへ
OPT.1 + Y	未定義	
OPT.1 + Z	#37	取り消し
OPT.1 + [	未定義	
OPT.1 + ¥	未定義	
OPT.1 +]	未定義	
OPT.1 + ^	未定義	
OPT.1 + _	未定義	

表6　デフォルトのファンクションキー設定

ファンクションキー		
DEL	#7	1文字削除
_	#4	カーソルを1文字右に移動
_	#19	カーソルを1文字左に移動
_	#5	カーソルを1文字上に移動
_	#24	カーソルを1文字下に移動
F1	#66	カーソルをファイルの先頭に移動
F2	#90	カーソルをファイル最終行に移動
F3	E74	カレントワードの後方置換
F4	E78	カレントワードの後方検索
F5	M1,'sea -N',$0D	次検索
F6	#40	範囲指定の開始，及び取消し
F7	#41	（カット）指定した範囲を削除⇒クリップボードへ
F8	#42	（コピー）指定した範囲　⇒クリップボードへ
F9	#71	クリップボードの内容をカーソル位置にn回複写
F10	#43	行の二重化
HELP	H10	ヘルプファイルを開く
UNDO	#59	キーボードマクロを実行する
CLR	#101	編集テキストの切り替え（降順）
HOME	#98	編集テキストの切り替え（昇順）
INS	#32	なにもしない
ROLL UP	#3	画面をロールアップ
ROLL DOWN	#18	画面をロールダウン
SHIFT + DEL	#7	1文字削除
SHIFT + _	#32	なにもしない
SHIFT + _	#32	なにもしない
SHIFT + _	#32	なにもしない
SHIFT + _	#32	なにもしない
SHIFT + F1	#70	新しいファイルの編集
SHIFT + F2	#79	テキストの編集を最初からやり直す
SHIFT + F3	E76	カレントワードの前方置換
SHIFT + F4	E83	カレントワードの前方検索
SHIFT + F5	M1,'sea -N -B',$0D	前方次検索
SHIFT + F6	#98	編集テキストの切り替え（昇順その2）
SHIFT + F7	#100	編集テキストの切り替え（降順その2）
SHIFT + F8	#89	ファイルを読み込み，カーソル位置に挿入
SHIFT + F9	#87	指定した範囲をファイルに書き出す
SHIFT + F10	#67	コマンドの実行
SHIFT + HELP	#217	キー定義の初期化
SHIFT + UNDO	#59	キーボードマクロを実行する
SHIFT + CLR	#32	なにもしない
SHIFT + HOME	#100	カーソルをホーム位置に移動
SHIFT + INS	#32	なにもしない
SHIFT + ROLL UP	#163	現在の選択位置を1画面下に移動
SHIFT + ROLL DOWN	#178	現在の選択位置を1画面上に移動

りますので，メニュー文字列は最短になるように書き直します。片仮名は半角化し，なくても意味の通る助詞を削除します。しかし半角片仮名は識別しづらいので，ほどほどにしておきましょう。

文字列の後ろについている“..”は，メニュー項目にさらに選択指定が加わることを表していますから，削除してはいけません。

でも階層メニューとリクエスタが混在しているのもなにか違和感はありますね。

文字列

シャーペンの定義コマンドとして記述できるものに文字列という項目があります。標準設定では外部ファイル名でしかつかわれていないのですが，テキスト中に直接文字列として挿入することができます。

例として拡張プレフィックス1を単文登録として使ってみましょう。

'中野　修一'　　* CTRL＋@ ＋A

のようになります。

これだと登録内容を覚えてないと使えないので，メニューに文字列を登録しておき，それを呼び出すことができれば……と考えます。しかし，単に，

'こんにちわ','こんにちわ'

のように定義しても（最初の文字列はメニュー内の見出し文字），

こんにちわこんにちわ

のような見出しの入ったメニューが開くだけでうまくいきません。

どうもメニュー文字列の直後には機能指定が要求されているようなので，

'こんにちわ',#32,'こんにちわ'

のように#32をはさんでやります。ちなみに，#32の機能は「なにもしない」です。

機能を併記してやればカーソルコントロールもできます。スタイル指定もできそうなのですが，少し問題もあります。

M1,'color -f6',$0D,'赤字'

だとなぜか2回に1回黒字になります。

#40,#4,M1,'color -f5',$1D,#40

のような指定もできます（1文字ごとに色を変えていく）。

ビットマップなどを指定すると「絵」になるのが残念ですが，単なる単文登録でも意外と用途が広いかもしれません。

謎のキーマップ変更

キーマップについては，いまひとつわからないところがあります。

割り当て変更は外部コマンドmapで行います。mapコマンドを使う場所は限られていますから，定義ファイルのE0の部分，もっとわかりやすくいえば，いちばん先頭で定義されているmapコマンドの内容を書き換えればよいことになります。

mapコマンドには2種類の動作が指定できます。

map -K

と，

map -M

です。

-Kオプションはいまいち使い方のわからない機能なのですが，一応，解説します。

理屈では，

map -K,255,$39,1,2

のように定義すると，SHIFT＋CTRL＋OPT.1＋OPT.2＋XF1＋XF2＋XF3＋XF4＋XF5＋9を同時に押したときにCTRL＋Bに定義された動作が行われるようになるはずです（押せねーって）。最初の“255”がシフトキーの指定，次の“$39”がキーコード，次の“1,2”でどの系統の機能の何番目を実行するかを指定します。

しかし，実際に指定してあるものを見ると自信がなくなります。

$2,$1b,$5,$40

と（$5は機能直接実行指定），

$0,$61,$4,$1f

です。それぞれ，

CTRL＋ESCが押されたら#64を実行
Aキーが押されたらS31を実行

という意味になるはずですが……。マニュアルの機能番号表は61～64の部分が誤っていますが，それを考慮しても不明点の多い記述です。

シフトキーの割り当てを変えるときに使うのは-Mオプションです。こちらはまだ明快です。ビット単位でシフトキーを指定し，マスクキーを指定し，機能の先頭位置を指定します。マスクキーに指定されていてシフトキーに指定されていないものが押されているときは動作は行われません。

デフォルト設定では，OPT.1，OPT.2，CTRL＋XF3，SHIFT＋CTRL＋XF3，CTRL＋XF4，SHIFT＋CTRL＋XF4の機能を順に定義しています。

ここで問題点です。たとえば，拡張プレフィックス2のデフォルト起動キー指定はSHIFT＋ESCとなっていますが，これは日本語入力時には使用できません（直前に変換した入力文字列が返される）。割り当てを変えたほうがよいでしょう。マニュアルによると拡張プレフィックスはmapコマンドで定義されているとなっていますが（少なくとも1は），見あたりません。

マニュアルの機能一覧には抜けていますが拡張プレフィックスは#64,#127の機能番号で実行されます。そのほか，外部コマンドのprefixを使っても処理できるのでそちらのほうがわかりやすいかもしれません。

なお，キーマップの変更を行った場合，キー定義ファイルを書き換えたあとはそれをキー定義として登録し，シャーペンに関するタスクをすべて閉じてシャーペンを立ち上げ直す必要があります。コンソールもシャーペンだということを忘れていると確実にハマリます。

その他

リターンキーの機能を改行挿入から，単に改行動作だけに変更しました。リターンキーを押しても行が折れ曲がらないという，要するにWP.Xと同じ動作です。遠い昔から即戦力ユーザーだった人には明快なのですが（そうでない人には改行挿入のやり方がわからない人もいたみたいですが）このような設定のときはSHIFT＋リターンで新しい行を追加できます。

特に日本語の文書を書いているときは改行挿入ではなく改行だけのほうが便利ではないでしょうか（いまいち自信なし）。

*　　*　　*

このようにシャーペンver.3.0は柔軟な設定が可能になっています。実によくできた処理系なのですが，最後に少しだけ次期バージョンへの要望を挙げておきたいと思います。

いろいろいじっていて感じたのは「変数がほしい」ということです。現在のカーソル位置とかが読み出せれば複雑な整形処理もできますし……，機能定義をマクロ言語に載せてしまえば完璧でしょう。

メニューマンを使わない自前のメニューを使っているのなら，メニュー文字列としてマルチテキストを使用できるようにしてほしかったところです。それから，MENU0だけでもシフト＋メニュー時の設定ができれば使いでがありそうです。

外部ファイルは本当に「なんでもできる」という印象を受けました。あとはサブルーチン的に呼び出されるのではなくて本体のイベントループ内で並列に動作するものも記述できれば無敵なのですが……。

というわけで，今後は誰かがLISPを載せるか，TeXのプレビュア化するかというのが楽しみでもあります。

究極のカスタマイズ

外部コマンドを作成する

Tamura Kento 田村 健人

シャーペンはユーザーによる拡張ができるようになっています
それが外部コマンドです
システムに用意されていない機能を追加してみましょう

たいていの仕事において，文書の作成という作業はあるものです。パソコンを実務に使いたいとしたら，文書の作成に使うのがいちばんです。

ゲームしかやらない人を除いて，パソコンを使ううえでかなりの時間を占めるのがエディタ(含ワープロ)での作業でしょう。仕事に使う文書，プログラムのソース，パソコン通信の書き込みなど，さまざまなものをエディタで書きます。このようにエディタはきわめて基本的なソフトなので，個人の好みがはっきりと出ます。UNIXではvi派とEmacs派に大きく分かれますが，X680x0ではED派とEmacs派ということになるでしょう。

さて，私はEmacs派で，うちではμEmacsかMuleを使っているわけですが，ちょっと困ったことがあったのです。SX-WINDOWにはμEmacsがありません。MuleはSX-WINDOW上でも動くのですが，起動するだけで1Mバイト以上のメモリを消費するのは少々きついものがあります。いや，そんなことよりも重要なのは，シャーペンがあまりに魅力的だということでしょうか。Emacsはワープロではありませんから，SX-WINDOWで軽快なEmacsが動いたからといってそれでいいというものではないのです。

「シャーペンで文書を入力したい，しかし私はEmacs派である」ということでキーバインドの変更に挑みますが，悲しいかな最初のシャーペンはEmacsの基本的なキーバインドを再現できるほど柔軟なカスタマイズはできなかったのでした。このことに関しては1993年5月号の瀧氏の記事に詳しく書いてあります。

SX-WINDOW ver.3.1になり，シャーペンのキーバインドのカスタマイズは恐ろしいほど柔軟になりました。これでEmacsライクなキーバインドもできるようになりました。

GNU Emacsの特徴として，Lispでプログラムを書くことにより機能拡張ができる点が挙げられます。GNU Emacsは基本的な機能以外はLispで書かれています。シャーペンも発表当時より，外部コマンドによって機能拡張ができるらしい，ということがいわれていました。

で，SX-WINDOW ver.3.1には外部コマンドの作り方がライブラリなども含めて収録されています。研究室のハードディスクを飛ばしたり，買ってきたメモリボードのコンデンサが取れてしまったりと，嫌なこと(苦笑)が続いていた私にとって，これはとても嬉しいことです。

外部コマンドを作る前に

私の記事は難解である，というのがもっぱらの評判ですが，この記事はそんなに難しくないです(たぶん)。

外部コマンドのプログラミングには，テキストマネージャに対するひととおりの知識が必要です。まあ今回掲載のサンプルプログラムがなにをしているのかがわかる程度でよいでしょう。

SX-WINDOW ver.3.0のテキストマネージャのコール名称を見ると，修飾のことを「スタイル」と呼んでいます。さらにヘッダのMTEXT.HというファイルやSX-WINDOW ver.3.0のシャーペンの呼び方から考えるに，修飾付きテキストの正式名称は「スタイル付きテキストエディット」か「マルチフォントテキストエディット」なのでしょう。片仮名だけで1行が埋まってしまいそうです。

外部コマンドでなにができるかを説明します。

1) テキストに対する任意の処理
2) 数値・文字列などの入力を促す
3) 一時的にイベントを乗っ取る

基本は1)のテキストに対する処理で，2)や3)は副次的なものと考えたほうがよいでしょう。たとえば文字列検索をしたいときには文字列の入力が必要になります。罫線を引きたいときはキーダウンイベントを乗っ取って「罫線入力モード」を実現するのです。

テキストに手を加えるだけの外部コマンドの動作は非常に単純です。外部コマンドは，シャーペンという親ルーチンから呼ばれるmain()という名のサブルーチンにすぎません。

疑似ダイアログなどを開いて数値・文字列の入力をするとなると，少々複雑な動作になります。理解するためのキーワードは「コールバック」です。シャーペンのライブラリにcommedit(), noedit()など疑似ダイアログなどを開く関数があります。これらを呼んだあとに外部コマンドのmain()関数を終了すると，シャーペンは疑似ダイアログが開かれたことを察知し，キー入力などは疑似ダイアログで行われます。改行キーや実行ボタンで入力が終了すると，commedit(), noedit()などを呼ぶときに指定した関数を呼びます。つまり，疑似ダイアログが開いている間は外部コマンドはなにもしないで，シャーペンが処理しているのです。(図1-b)

このように「あることが起きたらこの関数を呼んでね」という形式のことを「コールバック」といい，その関数を「コールバック関数」と呼びます。シャーペンの外部コマンドはウィンドウシステムとは特に関係はありませんが，コールバックはウィンドウシステムを高位ライブラリを用いて扱うときには必ずといってよいほど出てくる概念です。

イベントを乗っ取るというのは，前述した「罫線入力モード」や「上書きモード」を想像すればよいでしょう。一時的にしか乗っ取ることができないことに留意してください。

外部コマンドの作り方

外部コマンドの作成法に関してはシャーペンマニュアルにきちんと書いてありますし，役に立つサンプル，SHARPEN2.REFが拡張ディスクに入っているので，ここで詳しく解説するのは無駄だと判断しました。

基本的な作成法はマニュアルを参照してもらうことにして，ここでは外部コマンドを作成するにあたって私が気づいたことを列挙します。

SX-WINDOW ver.3.1の拡張ディスクの“¥ETC¥外部コマンド”と“¥福袋¥外部コマンド”内のファイルが紛らわしい。“¥福袋¥外部コマンド”にあるファイルが“¥ETC¥外部コマンド”にあるファイルを含んでいるようなので，“¥福袋¥外部コマンド”のほうを使います。

編集禁止のテキストに対して，内容を書き換えるSXコールを発行しても書き換わりません。しかし，セレクト範囲があったりすると表示が乱れることがあるようなので，やはり編集禁止かどうかはチェックしたほうが無難です。私は表示が乱れても気にしないので，自分で作った外部コマンドではほとんどチェックしていません。

セレクト範囲内のテキストを書き換えるときやカーソルを移動させるときは，処理前にTMDeactivate2()，処理後にTMActivate2()を呼びます。これを行わないと反転表示に矛盾が生じたり，カーソル点滅のタイミングで表示が変わったりします。盲目的に「そういうものである」と思って，必ず呼んでください。

set_keyなどで呼ばれるようにしたキーベウンイベント処理関数の引数のASCIIコードは，CAPSキーに関係なく単独で押されたキーは大文字，シフト併用で押されたキーは小文字となるようです。

テキストエディットレコードのメンバrefConを書き換えてはいけません。シャーペンが重要な用途に使っています。

SX-WINDOW上で外部コマンドをmake，実行する場合は気をつけなければいけません。一度シャーペンで外部コマンドを実行すると，その外部コマンドはメモリに置かれます。makeして新しくしても，シャーペンが終了しない限りは新しいものを読まずにメモリから読みます。新しく作り直したら必ずシャーペンを終了して，もう一度シャーペンを起動します。なお，コンソールもシャーペンであるということには注意が必要です。

ライブラリは最初にSHARPEN2.Aを指定してください。SXLIB.L，CLIB.L，libc.a，libsxc.aなどを先に指定してしまうと，まったく動かないものになってしまいます。

gccでコンパイルする場合，-SXオプションは絶対につけてはいけません。gccの-SXオプションはスタートアップが対応しているという条件で使えるものであり，外部コマンドのスタートアップは-SXに対応していません。外部コマンド内ではa5レジスタはシャーペン.Xのワークエリアを指すようになっています。

図1　外部コマンドの動作

a)　通常の場合

b)　疑似ダイアログを開く場合

スタートアップに依存した関数はまったく使えません。malloc()系統はまず間違いなく使えないはずです。libcのsprintf()などもスタートアップに依存しているようなので利用できません。

もちろんアセンブラでも外部コマンドを作れます。アセンブラでC言語の関数を作る方法がわかれば大丈夫でしょう。C言語で書けるプログラムでアセンブラで書けないものはありません。

サンプル

わりと単純な外部コマンドを2つ掲載します。両方とも，編集作業をシャーペンで行っている編集(U)氏のリクエストにより作りました。

●setkind1.ex

選択範囲中の1バイト文字のスタイルを変更します。

ベクトルフォントやらベジエフォントやらを使うと，1バイト文字がROMフォントになってしまい，格好悪いです。デスクアクセサリ集のフォントリンカでどうにかなるようですが，1バイト文字だけフォントの種類を変えられればいいわけです……というのが直接の動機ではありません。

コンピュータの画面の上では，多くの場合1バイト文字を「半角文字」，2バイト文字を「全角文字」という具合に，横幅を1：2の比率で表示することが多いと思います。しかしこれは，コンピュータの都合でしかないわけです。プリンタをお持ちの方ならわかると思いますが，たいていのプリンタでは1バイト文字と2バイト文字の横幅が3：4ぐらいになっています。

これがどのようなときに問題になるかというと，プリンタでコード印字を行うときです。新しいシャーペンではコード印字時にも字詰め設定が有効になるようになりま

サンプルの使い方

makeするにはgcc，has，hlk，make，libcなどが必要です。XCライブラリ用に書き換えるのは容易でしょう。リスト(Makefile setkind1.c，tohan.c)をすべて打ち込んでmakeしてください。一字一句間違えずに打てばできあがるというものではありませんので悪しからず。

setkind1.exとtohan.exができたら，それらをシャーペンのEXCOMディレクトリにコピーします。あとは適当なキーに割り当てるなり，メニューに登録するなり，OPT.1＋1で直接実行するなりしてください。

●1バイト文字のスタイル変更(setkind1)

書式　setkind1　[switches]

解説

setkind1は，選択範囲中の1バイト文字のスタイルを変更します。

-Fn　フォントフェイスをnにする

-Kn　フォントIDをnにする

nが省略されているときは入力を促します。

-Sx,y　フォントサイズをx,yにする

x,yが省略されているときは入力を促します。

書式例　setkind1 -F1 -K2 -S24,16

選択範囲中の1バイト文字を，強調，ROM24ドットフォント，サイズ24×16にします。

●2バイト文字を1バイト文字に変換(tohan)

書式　tohan

解説

tohanは，選択部分中の2バイト数字，アルファベット，記号を1バイト文字に変換します。選択されていないときはカーソル位置の文字を変換します。

した。試しに印刷してみましょう。結果は図2のようになります。右端がガタガタしていますね。これが半角文字のサイズの違いによる弊害です。

シャーペンのコード印字は印刷条件設定の項目で「文字間あり」にあわせてあるのです。これを使った印刷例を図3に示します(BJ-220C使用，文字間あり，行間5)。

好みにもよるでしょうが，これだとなんだか間延びして読みづらく感じられます。

この問題を解決するのが外部コマンドsetkind1.exです。このコマンドで，事前に1バイト文字の横幅を2バイト文字の3/4にしておけばコード印字時の印刷イメージにかなり近いものになります。たとえば16ドットフォントだと，1バイト文字の横幅は12ドットになれば幸いです。setkind1-s24,16とすればよいでしょう(横幅は2倍の値を与えなければならないことに注意)。

これを使用した印刷例を図4に示します。

プログラムではなにも難しいことはなく，選択範囲中で1バイト文字があったら指定されたスタイルに変更しているだけです。数値の入力を促す部分が参考になるかもしれません。

図2 コード印字(字間なし)

さて，私はEmacs派で，うちではμEmacs
かMuleを使っているわけですが，ちょっと
困ったことがあったのです。SX-WINDOWに
はμEmacsがありません。MuleはSX-WINDOW
上でも動くのですが，起動するだけで1Mバ
イト以上のメモリを消費するのは少々きつ
いものがあります。いや，そんなことより
も重要なのは，シャーペンがあまりに魅力
的だということでしょうか。Emacsはワー
プロではありませんから，SX-WINDOWで軽
快なEmacsが動いたからといってそれでい
いというものではないのです。

図3 コード印字(字間あり)

さて，私はEmacs派で，うちではμEmacs
かMuleを使っているわけですが，ちょっと
困ったことがあったのです。SX-WINDOWに
はμEmacsがありません。MuleはSX-WINDOW
上でも動くのですが，起動するだけで1Mバ
イト以上のメモリを消費するのは少々きつ
いものがあります。いや，そんなことより
も重要なのは，シャーペンがあまりに魅力
的だということでしょうか。Emacsはワー
プロではありませんから，SX-WINDOWで軽
快なEmacsが動いたからといってそれでい
いというものではないのです。

図4 コード印字(setkid1使用)

さて，私はEmacs派で，うちではμE
macsかMuleを使っているわけですが，
ちょっと困ったことがあったのです。SX-
WINDOWにはμEmacsがありません。M
uleはSX-WINDOW上でも動くのですが，
起動するだけで1Mバイト以上のメモリを
消費するのは少々きついものがあります。
いや，そんなことよりも重要なのは，シャ
ーペンがあまりに魅力的だということでし
ょうか。Emacsはワープロではありませ
んから，SX-WINDOWで軽快なEmacs
が動いたからといってそれでいいというも
のではないのです。

●**tohan.ex**

標準の外部コマンドcase.exと同じような動作で，2バイト文字を1バイト文字に変換します。

TEXを使ったことがある方にはよくわかると思います。2バイト文字の数字・記号・英字は印刷すると格好悪いのです。

不満点など

外部コマンドに関する不満を挙げてみます。

SHARPEN2.Aに入っている関数のプロトタイプ宣言を行うヘッダがありません。宣言がないわけですからコンパイラに注意されてしまうのです。無視すればいいのですが，関数名のタイプミスや引数の順番の間違いなどがコンパイル時に検出できないという不安があります。

また，MTEditをTEditにキャストするのが面倒です。(TEdit＊＊)ですから，シフトキーを6文字ぶんも押していなければなりません。WindowとGraphのように中に含まれるようにすれば楽になるんじゃないかと思うかもしれませんが，テキストエディットはハンドルで扱うので意味がないのです。いっそのことMTEditなんて型は作らないでTEditのままにしておいたほうがよかったのではないかと思います。

また，イベント処理を一時的に変更する手段しか用意されていません。各イベントで，恒久的にシャーペン本来の処理の前後に割り込んで処理できるような手段がほしかったところです。これができると外部コマンド同士の相性などが表れそうですが，シャーペンの可能性は一気に広がるでしょう。print2.exがページ枠を表示していることから類推すると，不可能ではなさそうです。が，方法が示されていないのは残念です。

ファイル入出力にフィルタをかけることはできないのでしょうか？　漢字コードの変換などをやりたかったところです。

外部コマンドや外部コマンド用ライブラリのマニュアルがインサイドSX形式でほしいですね。せっかくメーカー提供のオンラインマニュアルシステムがあるのですから，それを活用したいです。聞くところによると，市販ソフトにおいて紙のマニュアルのコストはけっこう大きいとのことです。

付録ディスク予告

この原稿を書いている時点で8つの外部コマンドを作っていて，今回掲載したのはそのうちの2つです。残り6つは付録ディスクに収録してもらう予定です。

Cのソースを入力するときに便利な「#関係」とか「/＊ ＊/関係」とか「インクリメンタルなにがし」。その他，自分で作って感動してしまった外部コマンドなどなど。

リスト1 setkind1.c

```
 1: /*
 2:         setkind1.ex  1Byteもじのスタイルを変更
 3:
 4:                                                   by けんと
 5: */
 6: #include <stdio.h>
 7: #include <ctype.h>
 8: #include <jctype.h>
 9:
10: #include <sxgraph.h>
11: #include <task.h>
12: #include <text.h>
13: #include <mtext.h>
14:
15: #include <sharpdf2.h>
16:
17: #define TMSelectP( ht ) ( (*ht)->selStart < (*ht)->selEnd )
18:
19:                                             /* 大域変数 */
20: TEStyle st;
21: int mask;
22: char buf[768];                              /* 汎用バッファ */
23: register edval* _val __asm( "a5" );
24:
25:
26:
27: void ChangeStyle( MTEdit** ht ) {
28:     long origst, origen;
29:     register long off;
30:
31:     if ( (*ht)->editMode & 0x10 ) {         /* 編集禁止のとき */
32:         return;                             /* 何もしない */
33:     }
34:     TMCacheFlush( (TEdit**)ht );
35:     TMDeactivate2( (TEdit**)ht );
36:                                             /* セレクト範囲を退避 */
37:     origst = (*ht)->selStart;  origen = (*ht)->selEnd;
38:
39:     off = origst;
40:     while ( off < origen ) {                /* もとのセレクト範囲でループ */
41:         int c;
42:         c = *(*(*ht)->text+off);
43:                                             /* 可視な1Byte文字であるか? */
44:         if ( 0x20<c && c<0x100 && !iskanji( c ) ) {
45:             long offch = off;
46:             do {                            /* 1Byte文字がどれくらい続いている? */
47:                 off ++;
48:                 c = *(*(*ht)->text+off);
49:             } while ( !iskanji( c ) && off<origen );
50:             (*ht)->selStart = offch;
51:             (*ht)->selEnd = off;
52:             TMChangeStyle( ht, &st, mask );     /* スタイル変更 */
53:         } else {
54:             if ( iskanji( c ) )  off ++;
55:             off ++;
56:         }
57:     }
58:                                             /* セレクト範囲を戻す */
59:     (*ht)->selStart = origst;  (*ht)->selEnd = origen;
60:     TMActivate2( (TEdit**)ht );
61:
62:     return;
63: }
64:
65:
66:
67: int 解釈_cb( MTEdit** ht, long code, long param ) {
68:     long l;
69:     int ret = 0;
70:     short* ps;
71:
72:     disposecomm( ht );
73:     l = TMGetText( (TEdit**)(_val->tehdl2), buf, sizeof(buf) );
74:     if ( l <= 0 )  return ret;
```

```
 75:     ps = (short*)(buf+700);
 76:     strtos( buf, ps, param );
 77:     switch ( param ) {
 78:       case 1:
 79:         st.font = ps[0];
 80:         break;
 81:       case 2:
 82:         st.size.p.x = ps[0];  st.size.p.y = ps[1];
 83:         break;
 84:       case 3:
 85:         st.font = ps[0];  st.size.p.x = ps[1];  st.size.p.y = ps[2];
 86:         break;
 87:     }
 88:     ChangeStyle( _val->tehdl1 );
 89:     return ret;
 90: }
 91:
 92:
 93:
 94: int GetOption( _LASCII param ) {
 95:     int i, ret = 0;
 96:     char c;
 97:     char* pc;
 98:     char** hc;
 99:
100:     st.option = 0;
101:     mask = 0;
102:     TSTakeParam( param, NULL, NULL, 2, (char**)buf, NULL );
103:     hc = (char**)buf;
104:     for ( i=*(int*)(hc)++; i>0; i-- ) {
105:         pc = *hc++;
106:         c = *pc++;
107:         if ( c == '-' ) {
108:             switch ( toupper( *pc++ ) ) {
109:
110:               case 'F':
111:                 mask |= 2;                          /* フォントフェイスの変更 */
112:                 st.face = atol2( pc );
113:                 break;
114:
115:               case 'K':
116:                 mask |= 1;                          /* フォントカインドの変更 */
117:                 if ( *pc ) {
118:                     st.font = atol2( pc );
119:                 } else {
120:                     ret |= 1;                       /* interactive: kind */
121:                 }
122:                 break;
123:
124:               case 'S':
125:                 mask |= 0xc;                        /* フォントサイズ縦横の変更 */
126:                 if ( *pc ) {
127:                     strtos( pc, (short*)&st.size, 2 );
128:                 } else {
129:                     ret |= 2;                       /* interactive: size */
130:                 }
131:                 break;
132:             }                                       /* switch */
133:         }                                           /* if - */
134:     }                                               /* for */
135:     return ret;
136: }
137:
138:
139:
140: int main( MTEdit** ht, long code, _LASCII param, long idexcom ) {
141:     int ret = 0;
142:     short* ps = (short*)(buf+700);
143:
144:     disposecomm( ht );
145:     switch( GetOption( param ) ) {
146:
147:       case 0:                                       /* 実行 */
148:         if ( !mask ) {
149:             st.font = G_ROM24;
150:             st.size.p.x = 24;
151:             st.size.p.y = 16;
152:             mask = 0xd;
153:         }
154:         ChangeStyle( ht );
155:         break;
156:
157:       case 1:                                       /* フォントIDを入力 */
158:         ps[0] = st.font;;
159:         stostr( buf, ps, 1 );
160:         commedit( _val->tehdl1, "     フォントID:", 解釈_cb, 1, buf );
161:         TMSetSelect( (TEdit**)(_val->tehdl2), 0, 0x7fffffff, 0x7fffffff );
162:         _val->fpthrough |= 2;
163:         break;
164:
165:       case 2:                                       /* 横、縦を入力 */
166:         ps[0] = st.size.p.x;  ps[1] = st.size.p.y;
167:         stostr( buf, ps, 2 );
168:         commedit( _val->tehdl1, "          横,縦:", 解釈_cb, 2, buf );
169:         TMSetSelect( (TEdit**)(_val->tehdl2), 0, 0x7fffffff, 0x7fffffff );
170:         _val->fpthrough |= 2;
171:         break;
172:
173:       case 3:                                       /* フォントID、横、縦を入力 */
174:         ps[0] = st.font;  ps[1] = st.size.p.x;  ps[2] = st.size.p.y;
175:         stostr( buf, ps, 3 );
176:         commedit( _val->tehdl1, "フォントID,横,縦:", 解釈_cb, 3, buf );
177:         TMSetSelect( (TEdit**)(_val->tehdl2), 0, 0x7fffffff, 0x7fffffff );
178:         _val->fpthrough |= 2;
179:         break;
180:     }
181:     return ret;
182: }
183:
184:
185:
186: int _command( MTEdit** ht, long code, _LASCII param, long idexcom ) {
187:     int ret = 1;
188:     switch( code & 0xffff ) {
189:       case 1:                                       /* メニューのチェックマークの作成 */
190:         ret = TMSelectP( ht );                      /* セレクト範囲があればOK */
191:         break;
192:       case 2:                                       /* 全テキスト終了 */
193:       case 3:                                       /* 編集中のテキストの終了確認 */
194:       case 4:                                       /* 編集中のテキストの終了 */
195:         break;
196:     }
197:     return ret;
198: }
```

リスト2 tohan.c

```
  1: /*
  2:         tohan.ex 1Byte文字にする
  3:                                                   by けんと
  4: */
  5: #include <stdio.h>
  6: #include <ctype.h>
  7: #include <jctype.h>
  8: #include <jstring.h>
  9:
 10: #include <sxgraph.h>
 11: #include <task.h>
 12: #include <text.h>
 13: #include <mtext.h>
 14:
 15: #include <sharpdf2.h>
 16:
 17:
 18:                                                   /* 大域変数 */
 19: char buf[768];                                    /* 汎用バッファ */
 20: register edval* _val __asm( "a5" );
 21:
 22:
 23:
 24: void ConvChar( TEdit** ht ) {                     /* 1文字変換する */
 25:     long origst, origen;
 26:     int c;
 27:     register long off;
 28:
 29:     TMDeactivate2( ht );
 30:     origst = (*ht)->selStart;  origen = (*ht)->selEnd;
 31:     off = (*ht)->selOffset;
 32:     if ( iskanji(  c = *(*(*ht)->text+off)&0xff ) ) {
 33:         c = (c<<8)+(*(*(*ht)->text+off+1)&0xff);
 34:         if ( !jiskata( c ) && !jishira( c ) ) { /* 仮名は除外 */
 35:             int l = 2;
 36:             if ( (c = zentohan( c ))&0xff00 ) {
 37:                 buf[0] = c>>8;
 38:                 buf[1] = c&0xff;
 39:                 buf[2] = '¥0';
 40:             } else {
 41:                 buf[0] = c;
 42:                 buf[1] = '¥0';
 43:                 l = 1;
 44:             }
 45:             (*ht)->selStart = off;
 46:             (*ht)->selEnd = off+2;
 47:             TMInsert( ht, buf, l );
 48:             TMCacheFlush( ht );
 49:         }
 50:     }
 51:     (*ht)->selStart = origst;  (*ht)->selEnd = origen;
 52:     TMSetSelCal( ht, off, off, off );
 53:     off = TMRightSel( ht );
 54:     TMSetSelCal( ht, off, off, off );
 55:     TMActivate2( ht );
 56:     return;
 57: }
 58:
 59:
 60:
 61: void ConvRegion( MTEdit** ht ) {                  /* セレクト範囲を変換する */
 62:     long origst, origen;
 63:     register long off;
 64:     long len = 0;
 65:
 66:     if ( (*ht)->editMode & 0x10 ) {               /* 編集禁止のとき */
 67:         return;                                   /* 何もしない */
 68:     }
 69:     TMCacheFlush( (TEdit**)ht );
 70:                                                   /* セレクト範囲を退避 */
 71:     origst = (*ht)->selStart;  origen = (*ht)->selEnd;
 72:     if ( origst >= origen ) {                     /* セレクトされていないときは1文字だけ */
 73:         ConvChar( (TEdit**)ht );
 74:         return;
 75:     }
 76:     TMDeactivate2( (TEdit**)ht );
 77:
 78:     off = origst;
 79:     while ( off < origen ) {                      /* もとのセレクト範囲でループ */
 80:         int c;
 81:         c = *(*(*ht)->text+off)&0xff;
 82:         if ( iskanji( c ) ) {
 83:             c = (c<<8)+(*(*(*ht)->text+off+1)&0xff);
 84:             if ( !jiskata( c ) && !jishira( c ) ) {
 85:                 int l = 2;
 86:                 if ( (c = zentohan( c ))&0xff00 ) {
 87:                     buf[0] = c>>8;
 88:                     buf[1] = c&0xff;
 89:                     buf[2] = '¥0';
 90:                 } else {
 91:                     buf[0] = c;
 92:                     buf[1] = '¥0';
 93:                     l = 1;
 94:                 }
 95:                 (*ht)->selStart = off;
 96:                 (*ht)->selEnd = off+2;
 97:                 TMInsert( (TEdit**)ht, buf, l );
 98:                 TMCacheFlush( (TEdit**)ht );
 99:                 off += 1;
100:                 len += 2-1;                       /* 何Byte減るか数える */
101:             } else                                /* 2Byte文字だけど1Byteにできない文字 */
102:                 off += 2;
103:         } else                                    /* 1Byte文字 */
104:             off ++;
105:     }
106:                                                   /* セレクト範囲を戻す */
107:     (*ht)->selStart = origst;  (*ht)->selEnd = origen-len;
108:     TMActivate2( (TEdit**)ht );
109:
110:     return;
111: }
112:
113:
114:
115: int main( MTEdit** ht, long code, _LASCII param, long idexcom ) {
116:     disposecomm( ht );
117:     ConvRegion( ht );
118:     return 0;
119: }
```

リスト3 Makefile

```
 1: CC =        gcc
 2: COPTIM      =       -O -fforce-mem -fforce-addr -fstrength-reduce
 3: CDEFS       =       -D_VARLEN=0
 4: CFLAGS      =       -Wall $(COPTIM) $(CDEFS)
 5: LD =        hlk
 6: LDLIBS      =       -l MTEXT.A SHARPEN2.A SXLIB.L libc.a libmb.a libdos.a
 7:
 8:
 9: all: setkind1.ex tohan.ex #xclick.ex i-f-comment.ex freturn.ex isearch.ex e-c-sharp-sign.ex
xpaste.ex
10:
11: %.ex: %.o
12:     hlk -o $@ $^ $(LDLIBS)
13:
14: %.o: %.c
15:     $(CC) $(CFLAGS) -o $@ -c $^
```

起動環境を整えよう

SYSDTOP.SXを斬る

Tamura Kento 田村 健人

ウィンドウの配置や画面まわりの情報を保存するSYSDTOP.SX 環境を変えてSX-WINDOWを立ち上げるなど非常時に有用なエディタを作成しました

私がX68000XVIの電源を入れていると，半分以上の時間はSX-WINDOW上で作業します。本誌の原稿はシャーペンで書いているので原稿執筆時にはほぼ100%SX-WINDOW上です。最近はさまざまなフリーソフトウェアのおかげでMAG/PI画像やPANICデータもSX-WINDOWを抜けずに見れるようになったので，SX-WINDOW上にいる時間はさらに長くなりました(各フリーソフトの作者様，ありがとうございます)。シャーペンのコンソールの影響も大きいでしょう。

エンドユーザーに徹して使うぶんにはSX-WINDOWの中身について知る必要はないかもしれません。しかし，万が一のときに備えて知識を貯めておくのも悪くありません。この記事でSX-WINDOW起動の要であるSYSDTOP.SXについて理解を深めましょう。

SYSDTOP.SXとはなんぞや?

SXWIN.Xと同じディレクトリ内に，SYSDTOP.SXがあります。このファイルを削除してSX-WINDOWを起動すると，ドライブアイコンなどはデフォルトの位置へ行き，ファイルアイコンやシンボルアイコンはデスクトップから取り払われ，ウィンドウは1枚もない状態になります。つまりSYSDTOP.SXというのはこれらの情報を記憶しているファイルである，ということがわかります。

「スタート画面設定」に「現在の画面を登録」「終了時の画面を登録」という項目があります。これらの「登録」を実行するとき，SYSDTOP.SXが更新されます。

この「登録」のときにデスクトップにいるタスクは次回起動時に同じ状況を作るためにそれぞれの方法で状態を保存します。さてここで，あるタスクの起動処理には不具合があり，条件によってはシステムエラーで落ちてしまうものとします。そして，「登録」のときに保存した状態がその条件にあってしまいました。このままSX-WINDOWを終了すると，次にSX-WINDOWを起動しようとしても起動途中でシステムエラーになってリセットせざるをえなくなってしまいます。問題のあるタスクを殺してもう一度登録をすればいいのですが，SX-WINDOWが起動しない限りタスクを殺すことも画面を登録することもできません。

この状況を打破するために，SYSDTOP.SXを消し去ります。たったひとつのタスクを殺すために，使い慣れたアイコンの配置，作業中だったウィンドウなども無理心中です。

実はもっと賢い対処方法もあります。問題のあるタスクの実行ファイルの名前を変えてしまうのです。この状態でSX-WINDOWを起動すると「〜を探しています」というダイアログが出るはずです。ここで「中止」押せばそのタスクは起動されません。

エディタで書き換えると自動で更新する

SYSDTOP.SXの内容

SYSDTOP.SXの中身について簡単に説明します。フォーマットの詳細については「追補版SX-WINDOWプログラミング」(吉沢正敏著)と今回掲載のesysd.cを見てください。「追補版〜」はSX-WINDOW ver.1.1当時のもので，デスクトップにファイルアイコンを置けるようになったSX-WINDOW ver.2.0からはSYSDTOP.SXのフォーマットが多少変更になっています。

SYSDTOP.SXは画面モード，タスク情報，アイコン情報，変数情報から成ります。

●画面モード

実画面モードのフラグは，80_Hのときはオフ，81_Hのときがオンです。

画面モードはDOSコールCRTMODの値と同じで，6万色モードのときは8が加えられます。

●タスク情報

各タスクの実行ファイルのフルパス名，コマンドライン，ウィンドウコンテンツ，グローバル座標からのオフセット，起動モード，リソースタイプとIDが収められています。

ウィンドウコンテンツとグローバル座標からのオフセットはどんな用途に使われるのか私には想像できませんでした。起動モードは，TSFock系のコールに渡す値です。リソースからの起動ではない場合，リソースの欄はCODE0080となります。

●アイコン情報

デスクトップに置き去りにされているアイコンの情報です。システムが使用しているアイコンとシンボルアイコンではリソースIDと座標，一般のファイルアイコンではフルパス名と座標が記憶されています。

さて，システムのアイコンとシンボルアイコンではリソースICN#のIDが記憶されるのですが，この値が8ビットで記憶されているのです。リソースIDはふつう16ビットですので，これでは全範囲をカバーできません。システムのアイコンのIDは1〜127，シンボルアイコンのIDは−2以下の負の値です。8ビットを符号つきとみなせば−128〜127となりますので，シンボルアイコンが−2〜−128の範囲ならば問題がないことになります。

で，がんばって127個のシンボルアイコン

を作ってみました。127個のシンボルアイコンが登録されている状態で新規のシンボルアイコンを作ろうとすると，アイコンメンテは「ICN# #—1」を編集しようとします。しかしこのアイコンは登録時にエラーになりました。よって，シンボルアイコンはリソースID—2〜—128の127個までという結論になります。

確認のため，リソースリンカを使って無理やりICN# —129を登録してみます。この状態でシンボルトレイを起動してもICN# —129は見えませんでした。

●変数情報

スクロールオフセット，ファイル検索パス，複数ファイルコピーのときの動作モードなどが，環境変数の定義のような形式で入っています。

esysd.x

SYSDTOP.SXを扱うプログラムのサンプルというわけではなく，この記事はesysd.xを紹介するために書かれたといったほうが正確です。esysd.xは，SYSDTOP.SXを編集するソフトです。

問題のあるタスクを殺すだけなら前述のファイル名変更でできますが，このソフトを使うほうが気分的に楽です(弱い理由)。アイコンの配置に凝りたいときにも便利かもしれません。

処理内容はたいしたものではなくて，SYSDTOP.SXをテキストに直してエディタを起動し，書き換えられたら再構築するだけです。ひたすらfprintf()とsscanf()です。

SX-WINDOW ver.3.0/ver.3.1のSYSDTOP.SXに対応していることを確認しています。ver.2.0のものに対して使えるかどうかはわかりません。

esysd.xの使い方

まず，gccがちゃんと動く環境が必要です。XCコンパイラではコンパイルできません。ライブラリ(とヘッダ)はlibcかXC2ライブラリです。SX-WINDOW関係のヘッダは必要ありません。C言語関係の設定がきちんと行われていれば，Makefileとesysd.cを打ち込んでmakeするだけで大丈夫です。大丈夫じゃないときは，設定が悪いということになります。

1)　起動する

a.下準備なし

引数にSYSDTOP.SXを指定します。

A>esysd a:¥SHELL¥SYSDTOP.SX

またはA>esysd a:¥SHELL

引数になにも指定しないときはカレントディレクトリのSYSDTOP.SXを対象にします。

b.下準備あり

環境変数SXSYSにSYSDTOP.SXがあるディレクトリを代入しておき，引数なしでesysd.xを実行します。

A>set SXSYS=a:¥SHELL

A>esysd

環境変数SXSYSが定義されているときにesysd.xに引数を与えると，引数で指定されたほうを処理します。

2)　待つ

SYSDTOP.SXの解析が終わるまで待ちます。ディスクの速さによりますが，数秒もかからないでしょう。

3)　エディタが起動される

SYSDTOP.SXの内容をテキスト化したものを読み込んでエディタが起動します。なにも設定していなければED.Xが，環境変数EDITORが定義されていればそのエディタを起動します。

設定例

A>set EDITOR=em.x

4)　眺める，もしくは書き換える

すでにある内容を書き換えたいときは常識を持って書き換えます。16進と書いてある項目はそのままで16進ですので，$とか0xとか&hはつけません。項目を増やしたり減らしたりするときは区切りの行も含めて編集し，項目の繰り返し構造を壊さないようにします。「こんな感じかな〜」で編集すればおそらく大丈夫です。

5)　セーブしてエディタを終了する

ED系ならESC E, Emacs系ならC-x C-s C-x C-c などでファイルをセーブして終了します。「やっぱやめた」というときはセーブしないで終了してください。

6)　待つ

編集結果からSYSDTOP.SXを再構築します。もし変なところを編集して明らかにおかしくしてしまっているときは，警告を出して終了します。なにも編集しなかったときはなにもしないで終了します。

7)　おわり

SYSDTOP.SXが新しくなっているはずです。もしesysd.xに不具合があってSYSDTOP.SXが正常な構造をしていなくても，再構築前のものがSYSDTOP.SXとして残っているはずですのでご安心を。

もうひとつの鬼門・スタートアップ

SYSDTOP.SXを抹消してもSX-WINDOWが起動途中で落ちるっっ，ということもあります。こういった場合はスタートアップメンテに登録してあるタスクの問題かもしれません。SYSDTOP.SXはスタートアップメンテに登録してあるタスクについては記憶しないのです。

SX-WINDOW以外でスタートアップメンテの中身を編集するのは簡単です。まず，RLK.X以外のリソースリンカを用意します。そのリソースリンカの使い方に従い，BUILTIN.LBのタイプShEVのID5番を抽出します。抽出したファイルをエディタで読み込んでみると，非常にシンプルな構造であることがわかります。臆せず編集して，元のBUILTIN.LB ShEV 5に戻せば完了です。

図1　スタートアップデータ(参考)

```
PROGRAM=sfile.? -S0
PROGRAM=A:¥SXSYS¥310¥SAdjust.r -Z816,544 -C$00,$16,$89,$0E,$1D,$83,$02,
$37,$05,$1E,$02,$3E,$1b,$ff,$28,$1a
PROGRAM=henwin.?
PROGRAM=A:¥sxbin¥sxmperiod.x
PROGRAM=A:¥sxbin¥sxmv.r
PROGRAM=A:¥sxbin¥vison¥extdrag.r
PROGRAM=sxcon.? -n
PROGRAM=ifm.x
PROGRAM=ivm.x
PROGRAM=A:¥sxbin¥SXerror.x -v
PROGRAM=A:¥sxbin¥2行にするの.r
PROGRAM=A:¥sxbin¥アイコン間隔.r
PROGRAM=A:¥sxbin¥SXkas.x 5b=5f 5f=5b
PROGRAM=A:¥SXSYS¥310¥SXKEY.X
PROGRAM=A:¥sxbin¥ClickMenu.x
PROGRAM=A:¥sxbin¥mscsr¥sxmscsr.x A:¥sxbin¥mscsr¥mscsr_b.cs
PROGRAM=A:¥SXSYS¥310¥SnapToforSX.r
PROGRAM=A:¥SYS¥HFONT¥FontExt31.x
PROGRAM=a:¥sxbin¥電卓管理.r a:¥sxbin¥数値演算.X
PROGRAM=A:/PROG/ISSL/EMDeCross_computer.r
PROGRAM=A:¥SXSYS¥310¥TSISRecToStr_slash.r
PROGRAM=a:¥sxbin¥コンピュータ画面.r
```

参考までに私の環境のスタートアップメンテの内容を掲載します。内容に関しては追求しないでください。

おわりに

本誌でSX-WINDOWの特集をやると，アンケートはがきにもっとも目につくのは「SX-WINDOWは使っていないので……」という言葉です。

SX-WINDOWはコマンドシェルに較べるとシャーペンとEasydrawがあるぶん実用になります。コマンドシェル上のソフトウェアは出尽くして淘汰されてきた感がありますが，SX-WINDOWはまだまだ未開拓の分野がたくさんあります。笑ってお仕事する環境，プログラミングの題材として，SX-WINDOWはいかがでしょうか。

SX-WINDOW起動の高速化

SX-WINDOW ver.3.1が届いたとき，久しぶりにすっぴんの状態のSX-WINDOWを立ち上げました。そこで驚いたのが，その起動の速さです。

前掲したように，私の環境ではスタートアップに20以上ものプログラムを登録しており，さらに常時10個以上のタスクがデスクトップに残ったままです。SX-WINDOW起動のバッチファイル(のようなもの。本当はzshのfunction)を実行してから，最初のアイドルイベントが発行されるまで，実に1分以上もかかってしまうのです(ウィンドウアイコニファイ，sxzcなどは最初のアイドルイベントから作業を始める)。システムディスクの環境ではスタートアップに4個のプログラムしか登録してありません。

こんな状態でもなんとかして起動を高速化できないか，と考えてみました。

たとえば，40Kバイトを20回に分けてメモリに読み込むのと，40Kバイトを1回でメモリに読み込むのでは，後者のほうが圧倒的に速いはずです。もうおわかりですね。スタートアップに登録してあるプログラムを，すべてCODEリソースとしてリソースライブラリにひとつにまとめます。これを一気にメモリにロードしてから各CODEリソースを実行するプログラムを作ってスタートアップに登録すればいいのです。

こういったことを自動でやるプログラム，誰か作ってみませんか? スタートアップのデータはTSRscGet2('ShEV',5,0)で得ることができます。TSPostEventTsk()でCHANGEDRSCイベントを発行してやればSX-WINDOW終了時にBUILTIN.LBを保存するかどうか聞いてくれる，という仕様も使えるでしょう。

リスト1 esysd.c

```
  1: #include <stdio.h>
  2: #include <stdlib.h>
  3: #include <string.h>
  4: #ifdef __LIBC__
  5: # include <jctype.h>
  6: # include <sys/stat.h>
  7: # include <unistd.h>
  8: #else
  9: # include <jfctype.h>
 10: # include <stat.h>
 11: # include <io.h>
 12: # include <process.h>
 13: #endif
 14:
 15: int Sysd2Temp( void );
 16: int Temp2Sysd( void );
 17:
 18:
 19: char fnSysdtop[FILENAME_MAX] = { 0, };
 20: char fnTemp[FILENAME_MAX];
 21: time_t ftTemp;
 22:
 23:
 24: int isnthkanji2( char* s, char* p ) {
 25:                                                  /* 文字列 s 中の *p の文字は ms-code
 26:                                                  の2バイト目であるか */
 27:     char* q = s;
 28:     while ( *q && q<p )
 29:         if ( iskanji( *q++ ) )  q ++;
 30:     return (q>p);
 31: }
 32:
 33: time_t getftbyname( char* fn ) {                 /* ファイルのタイムスタンプを得る */
 34:     struct stat st;
 35:     stat( fn, &st );
 36:     return st.st_ctime;
 37: }
 38:
 39: void MakeName( char* arg ) {
 40:     int l;
 41:     strcpy( fnSysdtop, arg );
 42:     l = strlen( fnSysdtop );
 43:                                                  /* パスの区切りで終わっているか */
 44:     if ( !isnthkanji2( fnSysdtop, fnSysdtop+l-1 )
 45:             && ( fnSysdtop[l-1] == '/' || fnSysdtop[l-1] == '¥¥' ) ) {
 46:         strcat( fnSysdtop, "SYSDTOP.SX" );
 47:     } else if ( stricmp( fnSysdtop+l-10, "SYSDTOP.SX" ) ) {
 48:         strcat( fnSysdtop, "/SYSDTOP.SX" );
 49:     }
 50:     return;
 51: }
 52:
 53:
 54: long main( int argc, char* argv[] ) {
 55:                                                  /* 1. sysdtop.sx のパスを得る */
 56:                                                  /* コマンドライン -> $SXSYS -> ./ */
 57:     if ( argc != 1 ) {
 58:         MakeName( argv[1] );
 59:     } else {                                     /* コマンドラインがない */
 60:         char* p = getenv( "SXSYS" );
 61:         if ( p ) {                               /* 環境変数がある */
 62:             MakeName( p );
 63:         } else {                                 /* コマンドラインも環境変数もない */
 64:             strcpy( fnSysdtop, "SYSDTOP.SX" );   /* カレントディレクトリ */
 65:         }
 66:     }
 67:
 68:                                                  /* 2. sysdtop.sx を読みつつ esysd.$$$ を作る *
/
 69:     fprintf( stderr, "%s を解析中¥n", fnSysdtop );
 70:     if ( Sysd2Temp() ) {
 71:         fprintf( stderr, "何等かの原因で一時ファイルを作成できませんでした¥n" );
 72:         return 1;
 73:     }
 74:                                                  /* 3. $EDITOR esysd.$$$ */
 75:     {
 76:         char* pe = getenv( "EDITOR" );
 77:         char bufc[256];
 78:         strcpy( bufc, pe ? pe : "ED.X" );
 79:         if ( spawnlp( P_WAIT, bufc, bufc, fnTemp, NULL ) < 0 ) {
 80:             perror( NULL );
 81:             fprintf( stderr, "エディタを起動できませんでした¥n" );
 82:             return 2;
 83:         }
 84:     }
 85:                                                  /* 4. esysd.$$$ が更新されていなかったら */
 86:                                                  /*    終わり */
 87:     if ( ftTemp == getftbyname( fnTemp ) ) {
 88:         fprintf( stderr, "更新しません¥n" );
 89:         return 3;
 90:     }
 91:     fprintf( stderr, "更新します¥n" );
 92:                                                  /* 5. sysdtop.sx のバックアップをとる */
 93:     {
 94:         char fnBack[FILENAME_MAX];
 95:         strcpy( fnBack, fnSysdtop );
 96:         strcat( fnBack, "~" );
 97:         unlink( fnBack );
 98:         if ( rename( fnSysdtop, fnBack ) < 0 ) {
 99:             perror( NULL );
100:             fprintf( stderr, "複製失敗¥n" );
101:             return 4;
102:         }
103:     }
104:                                                  /* 6. esysd.$$$ を読みつつ sysdtop.sx を作る *
/
105:     fprintf( stderr, "%s を再構築中¥n", fnSysdtop );
106:     if ( Temp2Sysd() ) {
107:         char fnBack[FILENAME_MAX];
108:         unlink( fnSysdtop );                     /* 作りかけを消す */
109:         strcpy( fnBack, fnSysdtop );             /* バックアップを元に戻す */
110:         strcat( fnBack, "~" );
111:         rename( fnBack, fnSysdtop );
112:         fprintf( stderr, "SYSDTOP.SX を作成できませんでした¥n" );
113:         return 5;
114:     }
115:
116:     return 0;
117: }
118:
119:
120: typedef struct {
121:     char name[90];                               /* ASCIIZ */
122:     char command[256];                           /* LASCII */
123:     short l_left;
124:     short l_top;
125:     short l_right;
126:     short l_bottom;
127:     short o_x;
128:     short o_y;
129:     char mode1;
130:     char mode2;
131:     long rsctype;
132:     short rscid;
133: } SaveTask;
134:
135: char* sMes[] = {
136:     "[実画面フラグ(16進)]:",
137:     "  [画面モード(16進)]:",
138:     "**========",
139:     "           [タスク名]:",
140:     "   [コマンドライン]:",
141:     " [コンテンツ(10進)]:",
142:     " [オフセット(10進)]:",
143:     "[mode1,mode2(10進)]:",
144:     "           [リソース]:",
145:     "*--------",
146:     "[ICN#ID(10進)]:",
147:     "        [Point]:",
148:     "[アイコン名]:",
149:     "     [Point]:",
150:     "[変数]:",
151: };
152:
153: int Sysd2Temp( void ) {
154:     FILE* pfs;
155:     FILE* pft;
156:     short ntask;
157:     long nicon;
158:     SaveTask st;
159:     short x, y;
160:     char iconname[FILENAME_MAX];
161:     char c;
162:
```


▶キーボードが壊れてしまった。修理から戻ってくるまではマウスキーボードの出番だぜ。
兼松 隆文(23)三重県

```
163:     if ( !(pfs = fopen( fnSysdtop, "rb" )) ) {
164:         perror( NULL );  return 1;
165:     }
166:     strcpy( fnTemp, fnSysdtop );                    /* SYSDTOP.SX と同じディレクトリに */
167:     strcpy( fnTemp+strlen( fnTemp )-10, "esysd.$$$" );
168:     if ( !(pft = fopen( fnTemp, "wt" )) ) {
169:         perror( NULL );  fclose( pfs );  return 1;
170:     }
171:
172:     fprintf( pft, "%s%2.2x¥n", sMes[0], fgetc( pfs ) );
173:     fprintf( pft, "%s%2.2x¥n", sMes[1], fgetc( pfs ) );
174:     fprintf( pft, "%s¥n", sMes[2] );
175:
176:     fread( &ntask, sizeof(short), 1, pfs );       /* タスク情報 */
177:     for ( ; ntask>0; ntask-- ) {
178:         fread( &st, sizeof(SaveTask), 1, pfs );
179:         fprintf( pft, "%s%s¥n", sMes[3], st.name );
180:         fprintf( pft, "%s%s¥n", sMes[4], st.command+1 );
181:         fprintf( pft, "%s%d,%d,%d,%d¥n", sMes[5],
182:                 st.l_left, st.l_top, st.l_right, st.l_bottom );
183:         fprintf( pft, "%s%d,%d¥n", sMes[6], st.o_x, st.o_y );
184:         fprintf( pft, "%s%d,%d¥n", sMes[7], st.mode1, st.mode2 );
185:         fprintf( pft, "%s%4s%4.4x¥n", sMes[8], (char*)&st.rsctype, st.rscid );
186:         fprintf( pft, "%s¥n", sMes[9] );
187:     }
188:     fprintf( pft, "%s¥n", sMes[2] );
189:
190:     fread( &nicon, sizeof(long), 1, pfs );       /* 名無しアイコン情報 */
191:     for ( ; nicon>0; nicon-- ) {
192:         fgetc( pfs );                             /* 定数0 読み飛ばす */
193:         fprintf( pft, "%s%d¥n", sMes[10], (signed char)fgetc( pfs ) );
194:         fread( &x, sizeof(short), 1, pfs );
195:         fread( &y, sizeof(short), 1, pfs );
196:         fprintf( pft, "%s%d,%d¥n", sMes[11], x, y );
197:         fprintf( pft, "%s¥n", sMes[9] );
198:     }
199:     fprintf( pft, "%s¥n", sMes[2] );
200:
201:     fread( &nicon, sizeof(long), 1, pfs );       /* アイコン情報 */
202:     for ( ; nicon>0; nicon-- ) {
203:         fread( &x, sizeof(short), 1, pfs );      /* 名前長さ */
204:         fread( iconname, sizeof(char), x, pfs );
205:         fprintf( pft, "%s%s¥n", sMes[12], iconname );
206:         fread( &x, sizeof(short), 1, pfs );
207:         fread( &y, sizeof(short), 1, pfs );
208:         fprintf( pft, "%s%d,%d¥n", sMes[13], x, y );
209:         fprintf( pft, "%s¥n", sMes[9] );
210:     }
211:     fprintf( pft, "%s¥n", sMes[2] );
212:
213:     fread( &nicon, sizeof(long), 1, pfs );       /* 環境変数情報 */
214:     while ( ( c = fgetc( pfs ) ) ) {
215:         fprintf( pft, sMes[14] );
216:         fputc( c, pft );
217:         while ( ( c = fgetc( pfs ) ) ) {
218:             fputc( c, pft );
219:         }
220:         fputc( '¥n', pft );
221:     }
222:     fprintf( pft, "%s¥n", sMes[2] );
223:
224:     fclose( pfs );
225:     fclose( pft );
226:     ftTemp = getftbyname( fnTemp );
227:
228:     return 0;
229: }
230:
231:
232: #define BSIZE   1024
233: #define RL()    if ( !fgets0( buf, BSIZE, pft ) )  goto inerr_exit
234: #define CL( N ) if ( strcmpz( sMes[(N)], buf ) )  goto inerr_exit
235: #define OL( N ) (buf+strlen( sMes[(N)] ))
236:
237: static __inline int strcmpz( char* p, char* q ) {
238:     return strncmp( p, q, strlen( p ) );
239: }
240: static __inline char* fgets0( char* p, size_t s, FILE* pf ) {
241:     char* t;
242:     t = fgets( p, s, pf );
243:     *strchr( p, '¥n' ) = 0;
244:     return t;
245: }
246: int Temp2Sysd( void ) {
247:     FILE* pfs;
248:     FILE* pft;
249:     long offc, offn;
250:     char buf[BSIZE];
251:     long l, m;
252:     short n;
253:
254:     if ( !(pfs = fopen( fnSysdtop, "wb" )) ) {
255:         perror( NULL );  return 1;
256:     }
257:     if ( !(pft = fopen( fnTemp, "rt" )) ) {
258:         perror( NULL );  fclose( pfs );  return 1;
259:     }
260:
261:     if ( !fgets( buf, BSIZE, pft ) ) {
262:   inerr_exit:
263:         fclose( pfs );
264:         fclose( pft );
265:         fprintf( stderr, "変なところ書き換えませんでした?¥n" );
266:         return 1;
267:     }
268:     CL( 0 );
269:     sscanf( OL( 0 ), "%x", &l );
270:     fputc( l, pfs );
271:
272:     RL();
273:     CL( 1 );
274:     sscanf( OL( 1 ), "%x", &l );
275:     fputc( l, pfs );
276:
277:     RL();
278:     CL( 2 );
279:
280:     offn = ftell( pfs );          /* short タスク数を書くところ */
281:     fseek( pfs, sizeof(short), SEEK_CUR );
282:     n = 0;
283:
284:     RL();
285:     while ( strcmpz( sMes[2], buf ) ) {
286:         SaveTask st;
287:         char* r;
288:         char* rr = NULL;
289:         memset( &st, 0, sizeof(SaveTask) );
290:
291:         CL( 3 );                  /* 名前 */
292:         strcpy( st.name, OL( 3 ) );
293:
294:         RL();
295:         CL( 4 );                  /* コマンドライン */
296:         strcpy( st.command+1, OL( 4 ) );
297:         r = strstr( st.command+1, " -w" );
298:         if ( !r )  r = strstr( st.command+1, " -W" );
299:         if ( r )  rr = strstr( r+3, " -" );
300: #if     0
301:         st.command[0] = (r&&!rr) ? r-st.command-1 : strlen( OL( 4 ) );
302: #else
303:         st.command[0] = strlen( OL( 4 ) );
304: #endif
305:
306:         RL();
307:         CL( 5 );                  /* コンテンツ */
308:         sscanf( OL( 5 ), "%hd,%hd,%hd,%hd", &st.l_left, &st.l_top, &st.l_right, &st.l_botto
m );
309:
310:         RL();
311:         CL( 6 );                  /* オフセット */
312:         sscanf( OL( 6 ), "%hd,%hd", &st.o_x, &st.o_y );
313:
314:         RL();
315:         CL( 7 );                  /* モード */
316:         sscanf( OL( 7 ), "%ld,%ld", &l, &m );
317:         st.mode1 = l;  st.mode2 = m;
318:
319:         RL();
320:         CL( 8 );                  /* リソース */
321:         sscanf( OL( 8 ), "%4s%hx", (char*)&st.rsctype, &st.rscid );
322:
323:         fwrite( &st, sizeof(SaveTask), 1, pfs );
324:         n ++;
325:         RL();
326:         RL();
327:     }
328:     offc = ftell( pfs );
329:     fseek( pfs, offn, SEEK_SET );
330:     fwrite( &n, sizeof(short), 1, pfs );
331:     fseek( pfs, offc, SEEK_SET );
332:
333:     offn = offc;                  /* long 名無しアイコン数 */
334:     fseek( pfs, sizeof(long), SEEK_CUR );
335:     l = 0;
336:     RL();
337:     while ( strcmpz( sMes[2], buf ) ) {
338:         short w[3];
339:         CL( 10 );
340:         sscanf( OL( 10 ), "%hd", w );
341:         w[0] &= 0x00ff;
342:         RL();
343:         CL( 11 );
344:         sscanf( OL( 11 ), "%hd,%hd", w+1, w+2 );
345:
346:         fwrite( w, sizeof(short), 3, pfs );
347:         l ++;
348:         RL();
349:         RL();
350:     }
351:     offc = ftell( pfs );
352:     fseek( pfs, offn, SEEK_SET );
353:     fwrite( &l, sizeof(long), 1, pfs );
354:     fseek( pfs, offc, SEEK_SET );
355:
356:     offn = offc;                  /* long アイコン数 */
357:     fseek( pfs, sizeof(long), SEEK_CUR );
358:     l = 0;
359:     RL();
360:     while ( strcmpz( sMes[2], buf ) ) {
361:         char fn[FILENAME_MAX+2+4];
362:         memset( fn, 0, FILENAME_MAX+2+4 );
363:         CL( 12 );
364:         strcpy( fn+2, OL( 12 ) );
365:         fn[1] = (strlen( fn+2 )+2)&0xfffffffe;
366:
367:         RL();
368:         CL( 13 );
369:         sscanf( OL( 13 ), "%hd,%hd", (short*)(fn+2+fn[1]), (short*)(fn+2+fn[1]+sizeof(short
)) );
370:
371:         fwrite( fn, sizeof(char), fn[1]+sizeof(short)*3, pfs );
372:         l ++;
373:         RL();
374:         RL();
375:     }
376:     offc = ftell( pfs );
377:     fseek( pfs, offn, SEEK_SET );
378:     fwrite( &l, sizeof(long), 1, pfs );
379:     fseek( pfs, offc, SEEK_SET );
380:
381:     offn = offc;                  /* long 変数総長 */
382:     fseek( pfs, sizeof(long), SEEK_CUR );
383:     l = 0;
384:     RL();
385:     while ( strcmpz( sMes[2], buf ) ) {
386:         CL( 14 );
387:         l += fwrite( OL( 14 ), sizeof(char), strlen( OL( 14 ) )+1, pfs );
388:         RL();
389:     }
390:     fputc( 0, pfs );  l += 5;
391:     offc = ftell( pfs );
392:     fseek( pfs, offn, SEEK_SET );
393:     fwrite( &l, sizeof(long), 1, pfs );
394:     fseek( pfs, offc, SEEK_SET );
395:
396:     fclose( pfs );
397:     fclose( pft );
398:
399:     return 0;
400: }
```

リスト2 Makefile

```
 1: #include=  /usr/include
 2: #lib      =       /usr/lib
 3: CC =      gcc
 4: CFLAGS    =       -Wall -O -fforce-addr -fforce-mem -fstrength-reduce
 5: MARIKO    =       ABF
 6: GCC_OPTION=
 7:
 8:
 9: esysd.x: esysd.o
10:     $(CC) -o $@ $^                      # libc のとき
11: #   $(CC) -o $@ $^ -lfloatfnc           # XC2 のとき先頭の#を外して上の行を削除
12:
```

▶X680x0用の「ストIIダッシュ」をプレイした友人が「久々にストIIらしいストIIをやった気がする」といたく感動しておりました。　　伴　武士(23)千葉県

華麗なるドット絵の世界
デスクトップを彩る

Nakano Shuichi 中野 修一

灰色だけではあまりにさみしい
たとえ制限された色数でも
きっとデスクトップを美しく彩ることができます

ウィンドウに色彩を

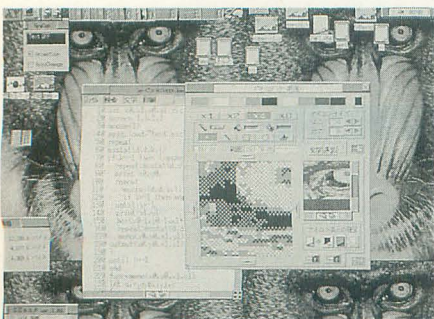
昔からどうもSX-WINDOWには色気が足りませんでした。

グレイスケールの画面は品がよいのですが，ディレクトリウィンドウが薄灰色の地なので，薄灰色をベースにしたアイコンを載せても沈んでしまいます。細かく書き込まれたアイコンはVS時代より，かえって情報量が低いような気さえしていました。

そもそも昔は，ウィンドウ環境で見た目の問題などは，リソース変更で簡単に解決することになっているものだと思っていたのですが，実はプログラム側が対応してなければどうしようもないんだということもうすうすわかってきました。

たとえば，シャーペンのメニューなんかもカラー化したいところなのですが，これはプログラムがPAT3データだと思って処理しているのでせっかく書き換えられるのにカラー化はできません。

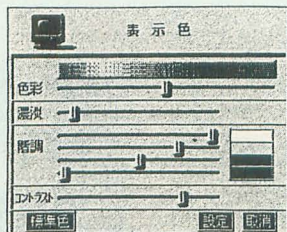
ディスクやファイルにある書き込み禁止マークもそうです。アイコン一覧には出てくるものは簡単にいじれますが，プログラムで呼び出しているものはPAT3データなのです。

また，リソースエディタなどが出てくればウィンドウの形などももっと自由にいじれるものと思っていましたが，ウィンドウ定義関数を扱うようなものはありませんでしたし，結局リソースというものの意義が半減しているような気のする今日この頃です。

カラーアイコンのすすめ

カラーアイコンというのは私が始めたわけではありませんし，むしろ最初はかなり疑問視していたものです。

カラーアイコンそのものならSX-WINDOW登場当時からいくつか見てきましたが，原色しか使えないためか，どれもケバくて幼稚に見えるという欠点がありました。それでもぽつぽつとカラーを使ったアイコンが増えていきます。すると，嫌でも，そのアイコンだけ妙に識別しやすいということに気づきます。

そうなるとあとは加速度的にカラーを使ったものが増えていき，一定量を超えたところで「全部やってしまえ」ということになります。編集部で使っているカラーアイコンは私や某編集氏が仕事の合間にいじったものです。

個人的な評価ですが，SX-WINDOWの純正アイコンはほかのウィンドウシステムのものと比べてもかなりよくできていると思います（明らかに駄洒落のような奴は勘弁してほしいものもありますが）。WINDOWSなどを見ると，16色とか256色とか使える環境なのに配色にデジタル色の文化が濃く出ているように感じられます。Macintoshはモノクロ機種との混在のためか平面的で単純なものが多いようです。NeXT STEPなどは趣味がよいのですが，いまひとつわかりにくく，文化の違いを感じさせるところがあります。

あまり独自のものを作ってもわかりづらくなるだけですので，デフォルトのものを基調としてエディットを進めていきました。方針は「できるだけ彩度を下げる」ということでした（これでもケバイという方もいましたが）。

なお，アイコン作成時のモノトーンパレットは極端に低コントラストに設定されていました。ついでにいえば，カラーパレットもグレイスケールと親和するように低彩度に抑えてあったのですが，まあ，これは好みが分かれるところかもしれません。

なお，システム色設定を低コントラストに仕上げると，画面のギラつきを抑えるとともにインタレース時のちらつきを抑える効果もあります。どの程度に設定されていたのかというと図1-Bくらいまでです。

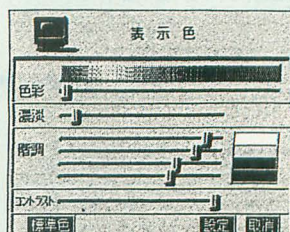
そのほかグレイスケールは少しだけ赤味をつけておくとひなまつり版カラーアイコンのエディット時の設定に近くなります。

タイリング

結果的に色数が増えるのですから「カラー化すれば情報量を上げることができる」というのは当たり前のことなのですが，生

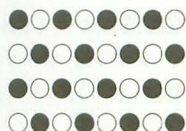
PATAデータに変換

図1-A デフォルトのパレット設定

図1-B 低コントラストのパレット設定

のままの原色はよほどのことがないとグレイスケールとは親和しません。

いずれにせよ，まともな色は4色しかありませんので，タイリングするしかありません。できれば，なにも知らない人に「4色しかない」ことがばれないようなものということで，できるだけ見かけの色数が多くなるようにしたいところです。

しかし原色同士というのはなかなか親和しないのです。いろいろやってみると，使用に耐えるのはAA55タイプのタイルパターンだけのように思われました。AA55というのは，

●○●○●○●○
○●○●○●○●
●○●○●○●○
○●○●○●○●

という感じの互い違いの配列です。

誤差拡散を使えば，確かに自然な階調変化を表現できるのですが，ほかのウィンドウアイテムとの質感のバランスが崩れてしまいます。取り込み画像などにはまだいいんですが，ペイント系で描いたCGを使ったりすると非常にノイジーになりますし，なによりパターンエディタでのドット修正が難しいのです。

大きめのディザパターンはエディットが大変なわりに効果が薄くて報われません。AA55タイプなら斜めラインだけで簡単に作成できます。

モノクロタイル変換プログラム

すでにユーザーレベルでたくさんのアイコンが出回っていますが，まだモノトーンのみのものが多かったり，カラーの場合はいまだに原色過多の傾向が強いように感じられます。

もともと，モノトーンだけだったシステムのアイコンとの質感の違いも気になります。そういうときには，最初はモノトーンで作成しておいて色を加えていくというのは有効な方法となります。

ということで，カラー画像からモノクロ画像（7階調）への変換プログラムを紹介しましょう。これは「X-BASIC」のプログラムです。SX-BASICではありませんので注意してください。

このツールではG-RAM上の画像をマウスで切り出してタイリングを用いたPAT4形式のモノクロ画像ファイルを作成します。

で，変換結果としてどのようなアイコンが作れるのかというと，すでに皆さんはご存じのはずです。3月号の付録ディスクに，カラーアイコンに交じって入っていたスズメとかアジサシとかはこういうプログラムで作られたものだったわけですね。

さて，プログラムの解説を行いましょう。PAT4ファイルの構成は単純です。先頭に00Hを4バイト分のヘッダ，ワードサイズでX，Yのサイズ，あとはプレーンごとのデータが並ぶだけです。

使用するときはプログラムに直接埋め込まれたファイル名などを適当に処理してください。

＊　＊　＊

カラーアイコンのなかに，シリーズものとして「カンバスのなかに絵を描く」というのがありました。結構頑張って作っていたのですが，ネタ切れで結局いろんな画像フォーマットの数だけ揃えることができませんでした。

いくつかは元絵がわからない人もいると思います。

難しそうにも見えますがドット数が少ないということは，失敗する部分も少ないということで，デッサン狂いの修整などは実に手軽にできます。これはドット打ちやアイコンエディットの練習課題としてはなかなか面白い題材ではないかと思います。ディレクトリ内にドット絵のギャラリーを開くというのもまたオツなものです。

リスト1

```
 10 int x0,x1,y0,y1,z,c,d,i,j,k,l
 20 screen 1,3,1,1
 30 mouse(1)
 40 apic_load("test.pic")
 50 repeat
 60 msstat(d,d,k,l)
 70 if k=-1 then { mspos(x0,y0)
 80   repeat:msstat(d,d,k,l):until k=0
 90   print x0,y0,
100   repeat
110     msstat(d,d,i,l)
120     if i=-1 then mspos(x1,y1)
130   until i=-1
140   print x1,y1
150   box(x0-1,y0-1,x1+1,y1+1,44444)
160   repeat:msstat(d,d,k,l):until k=0
170   mono(x0,y0,x1,y1)
180 output(x0,y0,x1,y1)
190 }
200 until l=-1
210 end
220 func mono(x0,y0,x1,y1)
230 int q,r,g,b,i,j,c
240 for i=y0 to y1
250   for j=x0 to x1
260     c=point(j,i)
270     b=(c shl 26)shr 27
280     r=(c shl 21)shr 27
290     g=c shr 11
300     q=(g*11+b*2+r*7)/96   :/* 96=(11+2+7)/7*32 → 7階調に
310     pset(j,i,rgb(q*5,q*5,q*5))
320   next
330 next
340 endfunc
350 func output(x0,y0,x1,y1)
360 str f
370 int x,y,i,k,x2,x3,a,b,c,d,e
380 input "ファイル名:";f
390 k=fopen(f,"c")
400 fputc(0,k):fputc(0,k):fputc(0,k):fputc(0,k)
410 x2=(x1-x0)*4/3+1                :/* 横何ドット分か?
420 x3=(x2+15)/16                   :/* 横何ワードか?
430 fputc(x2/256,k):fputc(x2 mod 256,k)
440 fputc((y1-y0+1)/256,k):fputc((y1-y0+1)mod 256,k)
450 for i=0 to 1
460   for y=y0 to y1
470     print y-y0
480     for x=0 to x3-1
490       e=0
500       for a=0 to 15
510         b=x0+x*16+a         :/* b=x座標(出力時)
520         c=point(x0+(x*16+a)*3/4,y) shr 11
530         c=6-c/5
540         if (c mod 2 =1)and((b+y)mod 2=0) then c=c+1
550         if i=0 then d=(c+1) mod 2 else d=c/4
560         if i=0 and (c=0 or c=5 or c=4) then d=0
570         e=(e shl 1)+d
580       next
590       fputc(e shr 8,k):fputc(e mod 256,k)
600     next
610   next
620 next
630 for i=1 to x3*2*(y1-y0+1):fputc(0,k):next
640 for i=1 to x3*2*(y1-y0+1):fputc(255,k):next
650 fclose(k)
660 endfunc
```

より美しい表示を求めて

メガディスプレイ追記編

Taki Yasushi 瀧 康史

ウィンドウ環境を快適にする広い画面
ハードの壁を越えてみましょう
そこにはメガピクセルの世界が広がっています

よりよい映りを求めて

私は6月号で，とにかく映りをよくするためにいろいろとCRTCの研究をしました。マルチスキャンディスプレイとオシレータ交換による高解像表示実験によりSX-WINDOWは初めてフルスクリーンでのオペレーションが可能になり，その真価を発揮できるようになったと思います。自分では最良にまとめたつもりだったんですが，記事が長すぎたせい(これもひとえに私の能力不足です。ごめんなさい)で，書き落としとか説明不足とか，そのほかの不明瞭な点をいくつか残したままになってしまいました。

これではいささか記事としては不十分ですので，今回，新たにページをもらいまして，上記の分，そしてその後調べた分を，ここに記しておきましょう。

8月号のDōGAの付録ディスクに，CRTC.Xという，モニタの解像度をいじって遊ぶには最適な玩具が入っています(Taka2さんありがとう)。自作ツールにあんまり変なモードを入れてバラまいた場合，細かい問題がいろいろと生じますが，それは置いといて，X680x0はこんなこともできるんだ！　と自分で納得するまでいじってみることにしましょう。

それから，一部初心者でも遊べるところはありますが，この記事はかなりマニア向けです。最低でも6月号のメガディスプレイの記事は読んでから，これ以降を読み始めてください。

表示幅はどこで決定するか

まず，表示幅についてです。パラメータについていろいろ説明しましたけれど，実際，画面の表示している部分については，前回触れていませんでした。

FM TOWNSを，X680x0シリーズのモニタに映したことがある方は，どの程度いるでしょう？　どちらのマシンも，15kHz～24kHz～31kHzという水平同期を持っていますが，FM TOWNSをX680x0モニタに映すと，左右が伸びすぎて画面からはみ出て切れてしまい，上下が逆に小さくなりすぎて，横長になってしまいます。逆にFM TOWNSのモニタにX680x0を映すと，デフォルトの画面では縦長になってしまいます（CRTCをいじればTOWNSに近いモードも作れます)。

どちらがよくて，どちらが悪いというわけではありませんが，FM TOWNSは垂直同期パルス幅，水平同期パルス幅，水平バックポーチ，フロントポーチなどが短いようです。それはさておき，実際のモニタの中で画面を表示している部分は，どのように決まっているのでしょうか？

それは，ブランキング時間です。

基本的に，水平同期期間は，水平データ表示期間＋水平ブランキング期間です。このうち，水平データ表示期間は実際に1ラスター横にラインを引く時間で，水平ブランキング期間は画面に表示していない時間です。画面の描画は，左上から書き始めて右上に移行し，1ライン下の左に戻ります。そう。画面の表示幅のサイズは，水平ブランキング期間，水平データ表示期間の比率で決まるのです。

ここである実験をしてみましょう。

8月号のオマケディスクに入っていたCRTC.Xを起動し，水平ブランキング期間を減らしていきます。水平ブランキング期間というのは，水平同期パルス幅，水平バックポーチ，水平フロントポーチの和ですから，上記のパラメータのどれをいじっても水平ブランキング期間は減ります。

簡単に説明しましょう。水平同期パルス幅はブラウン管の電子ビームの発射銃が右から左に戻る時間（実際はもちろん機械的な銃なのではなく，電磁波によって曲げられているだけである)，フロントポーチは，モニタに映した左の余白(黒？)，バックポーチは右の余白です。

(0，0)ドットの部分から電子銃の移動を追うならば，水平データ表示期間→水平フロントポーチ→水平同期パルス幅→水平バックポーチというところですか。このあたりは6月号で細かく説明したので，あちらを読んでください。

さあ，ブランキング期間を少なくするとどうなりましたか？　徐々に垂直同期が上がる分，画面の瞬きが少なくなり，代わりに画面が左右に広がります。あなたのモニタに根性があるならば，目に見えて画面が広がる様子がわかると思います。

同じドットクロックでより多くの表示を求めるということは，逆に，これらのブランキング期間をケチるということです。ブランキング期間をケチれば逆に画面は左右に広がります。この絶妙なバランスのなかでCRTCの設定を楽しんでください。

垂直同期とシステムタイミング

前回，ちょこちょことハードウェアの改造について書きました。その内容はCRTCに供給されているクロックを69MHzから100MHzのオシレータに上げるというものです。

これはなかなかヘビーな改造で，それなりのリスクがつきまといます。その代わり，効果は大きいので，ハイリスクハイリターンの典型といったところでしょうか。

前回は，自分自身が改造を行って間もない時間に発表したので（それでも1カ月以上寝かせましたが，それなりに寒いときでしたし)，安全面において原稿に不完全な部分がありました。

まず，いちばんのハイリスクは，クロックを上げたことにより，内蔵ICがついてい

かないことがあるということです。

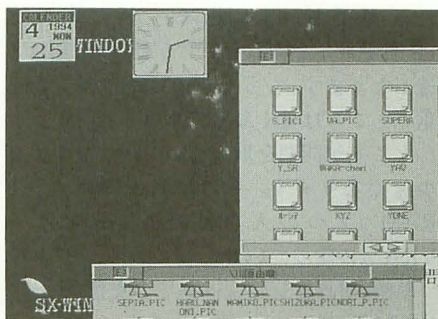
「InsideX68000」などのシステムブロック図を見て見ればわかるのですが，ドットクロックオシレータを変えたことにより，影響を受けてしまうICは，まずOSCIAN(2)，VICON，VIPSなどです。あと，SONYのビデオ用コントローラがありますが，これも大きさがかなり小さく，またかなり熱を持ちます。

まずOSCIANですが，これがついてこないということはまずありません。問題はVIPS，VICONといったところで，これらはモロに影響を受けます。これらが熱くなってくると画面が乱れ始めてきます。具体的には，テキストと，グラフィックの合成に失敗したりします。SX-WINDOWの背景設定で，16×16ドットの透明色となんらかの色のメッシュ（市松模様）を作ってみましょう。モニタのブラウン管の高低によって，モアレ縞が出ていますが，これは無視です。

このあとGRWウィンドウを出し，これにグラフィックを表示すると，透明色の部分で後ろのグラフィック画面が透けて見えます。メッシュがかかる分，使った色により暗めに見えたり，明るめに見えたりしますが，これで熱が本体内にこもり始めてくると機種によっては画面の至るところで画像が崩れてくるのがわかります。

ちょうど，グラフィックとテキストが画面の合成に失敗しているようなのです。これを今月のローテクのコラムで使用した強制冷却スプレーを使ってVIPSなどを冷やすと，一気に画面が落ち着きます。そして，画面に異常が起きている状態で放っておくと，CRTCが動作しなくなり(CRTCの熱暴走）画面に映らなくなります。さらに放っておくと，最悪の場合CRTCが熱破壊をするかもしれません。

これを防ぐためにはどうすればいいのでしょうか？　VICONとOSCIAN，VIPSはいずれも上になんらかのボードが差さる部分にいます。たとえば，OSCIANやVICONは拡張スロットの下ですし，VIPSはXVIなどではむき出しの場所にありますが，X68030の場合は，CRTアンプボックスの真下にいます。

そのため，ヒートシンクをつけるにしても，かなり薄めのものが必要です。VICONはそれほど熱を持つほうではありませんが，386CPU(486DLC,486DRx2)用の薄型ヒートシンクをつけることをおすすめします。ファンを使って風をうまく当てることができないのが残念ですが，なんとかつけてうまい空気循環を考えてください（詳細はローテクのコラム参照)。

また，VIPS，OSCIAN2はかなり小さいICなので，薄型で小さめのヒートシンクを物色しなければならないかもしれません。

画面が乱れる最大の犯人はVIPSです。これに風を当てられるようにうまくヒートシンクをつければ，かなり安定します。

これらが最大のハードウェア的なリスクですね。

さて，すでに改造を行ってしまった人はわかると思いますが，ソフトウェア的にも問題があります。

ここで具体的に例を出すことにします。

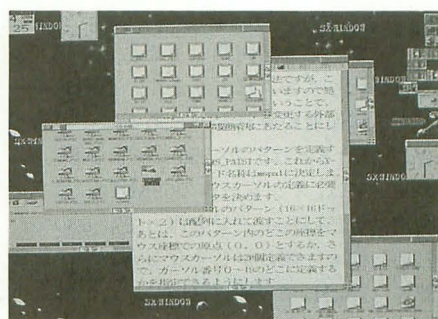
まず，69MHzのオシレータを100MHzに取り替えた24MHz改造XVIを例にしてみましょう。

100MHzへのクロックアップにより，画面がかなり綺麗に映ることになりますが，ソフトウェア的なリスクがつきまといます。

まず69MHzのときに，768×512の一般的なモードについて考えます。これは水平同期31kHz，垂直同期55Hzというモードです。純正ディスプレイは短残光といいつつも，多少残光しているようなので，垂直同期55Hzではちらつきは感じません（1秒間に55枚画面を更新している)。ところが，最近のハイレゾモニタは，見事に短残光なので，55Hzでは結構ちらちらします。

これを100MHzにクロックアップすると，どうなるのでしょうか？　これは比率的に69：100＝31.5：XでXは45.5kHzぐらいになります。水平同期はどうでもよいのですが，垂直同期はどうなるでしょう？　計算では，なんと79.7Hzになってしまいます。

79.7Hzならばもはやインタレースでもそれほどちらつかない周波数です。1秒間約80枚も書いているのですから，ちらつきはほとんどありません。シメシメと思いたいところですが，ここに落とし穴があります。

まず垂直同期が上がっているということ。これはゲームやアニメーションなどにとっては致命的です。一般にゲームやアニメーションは，垂直同期でタイミングを取ってスピードをコントロールしています。つまり，ゲームやアニメーションが100÷69≒1.45倍速くなるということです。これは目に見えるスピードアップです。

そのほうが面白いじゃんと思うかもしれませんが，そう気楽なことはいってはいられないところもあります。プログラムによっては，垂直同期の割り込みが嫌なところで入ることがありえるわけですから，場合によっては完全に戻ってこない暴走をします（例：あすか120%)。ジオグラフシールとかは1.45倍速で結構楽しいのですけどね。

さらに嫌なのは10MHzマシンの場合です。いまの例は24MHzにクロックアップしたマシンですから，単純計算で，2.4倍の処理速度があります。スピードが1.45倍速くなるということは，その分，1.45倍以上の処理速度を求められているわけでして，それがない場合，次の垂直同期を待つことになります。結果として，100÷(60×2)＝0.72という具合に，3割近くもゲーム速度が落ちることになるのです。

速度だけではありません。

ものによっては，インテリジェントジョイスティックのコントロールも垂直同期で見ているようで，メガドライブパッドを利用したゲームは一部コントロールできなくなります（例：ストリートファイターIIダッシュ)。

ゲームメーカーを責めてはいけません。一般に，ゲームの速度が垂直同期を使って同期するのは，画面の描画タイミングで画面中にノイズを出さないためで，我々はその要となるタイミングをいじってしまったのだから，おかしくなって当然なのです。

ゲームはしないからいいやと思う方もいるかもしれませんが，そうはいっていられないところでときどき，垂直同期を見ています。

まずは音楽データと同期するもの。

たいていの音楽ピュアなどはこけます。

そして，アニメーションデータなどは，

PCM音と画面が思いっきりバラバラになってしまって全然同期しません。

ほかにもリスクはあります。15kHzモードで、解像度がそれなりにあるモードの場合です(たとえば512×256モードなど)。このモードは,オシレータの69MHzのほうを利用して15kHzモードを作っているので,15kHz側も一部おかしくなるわけです。計算では15kHzモードは20kHzぐらいになり,垂直同期も80Hzぐらいまで上がります。

となると69MHzマシンの場合,かなりいろいろなところでリスクが出てきます。SX-WINDOW専用マシンならば適当に処理できますが,ゲームなどができなかったり,SX上でCGAファイルがものすごいタイミングで再生されたりしますし。

いちばんよいのは,69MHzのほうではなく,コンパクトXVI以上に限定されてしまいますが,50MHzのオシレータを100MHzに取り換えることです。このオシレータは,あとから出てきた分,使われていないので,安心といえば安心です。

ただ,このオシレータは回路図からわかる通り,1ピンでコントロールされており,これには付加回路が必要になります。回路さ図1に掲載しておくので参照してください。

ブランキング期間とメモリウェイト

X68000のゲームは,なぜ15kHzにすると速度が上がるのでしょう?

答えは簡単です。

それは,垂直同期の違いです。31kHzモードは垂直同期が55Hzなのに比べ,15kHzモードでは,垂直同期が60Hzになるからです。このため,垂直同期できちんとタイミングを取ったゲームなどは15kHzモードにすると,だいたい1割ほどスピードアップします。

某10Mショックシリーズの筆頭と謳われた沙羅曼蛇も15kHzモードにすると(HELPキーを押しながら起動),1割ほど速くなります。もっとも10MHzマシンだと重いところは重くなりますけどね。

ところが同期を取っていないゲームも多少速くなります。それはデュアルポートRAM(2つの入出力のあるRAM)をVRAMに使っているためです。

VRAMはコンピュータとの交信,つまりVRAMを書いたりするようなメモリアクセスと同時に,画像として出力しなければいけません。画像出力は画面に書いている期間,つまり水平表示期間の間です。大昔のパソコンはデュアルポートRAMを使っていなかったので,パソコンはブランキング期間しか,VRAMに書き込みができませんでした。確かPC-8801mkIIまでのPC-8801シリーズはそういう感じだったと思います(PC-8801はSRでデュアルポートになったんだったかな?)。

最近のマシンはデュアルポートRAMを利用しているため,表示期間もVRAMの内容をいじることができます。いじることができますが,コンピュータから見て,デュアルポートRAMのもう片方のI/Oがアクセス中,つまり表示期間と非アクセス中(ブランキング期間)ではメモリのウェイトが異なるのです。

要するに,ブランキング期間が多いCRTCモードであればあるほどVRAMはプログラムから見て軽くなります。15kHzモードは31kHzモードに比べ,水平同期期間が長くなっています。その分,ブランキング期間が長いので,VRAMのウェイトが少ない時間が長いのです。

結果として,15kHzモードのほうが同期を取っても取らなくても速いことになるわけです。

ところで,先ほどの改造マシンですが,100MHzに変えてしまったら,当然ブランキング期間も短くなります。しかし,表示期間も実際には短くなりますから,結果的にはあんまり変わらないかもしれません。

図1

純正モニタの臨界点

ここらで,純正モニタに話を移してみましょう。純正モニタではいったいどのくらいまで映すことができるのでしょうか?ちょっと気になったので試してみました。ただ,私の身の周りにないモニタまではわかりませんでした。また,これは,多くのモニタのロットの中から,たったひとつだけ私がサンプリングした結果ですから,個体差でほかのモニタで動かない可能性もあります。

すべてのデータはSX-WINDOWなどにあわせて作成されています。

これを探究した知的所有権が私につきそうな気もしますが,これは実質的に放棄することにします。どうぞご自由に活用してください。

●CZ-6xx系(3モードオートスキャン)

CZシリーズのモニタの長所は,垂直同期パルス幅が短くても同期が取れることです。特殊なCRTモードを使ったソフトを作るとき,このモニタだと垂直同期パルス幅を1ラインまで減らしても追従します(もっとも,最近のモニタは当然のように追従しますが)。

わりと映りがよく,グラフィックを見てもベタっとした感じがして気にいってはいるのですが,文字を映すモニタじゃないかもしれません(いまとなっては)。

さて,このモニタでいろいろ探究してみました。

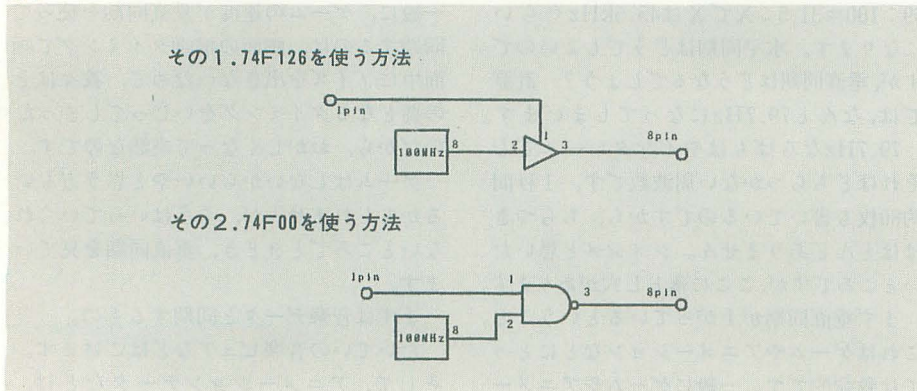
CRTC.Xで探究したところ,だいたい水平は32.5kHz,垂直は67Hzぐらいまで追従できるようです。

たくさんロットがあるので,ロット差が大きく,ものによっては垂直同期は67Hzが追従しないものもあります。実際の値は表1を参照してください。ただ,たいていのものは63Hzぐらいまでは追従するようなので3番を用意してみました。

インタレースは基本的に,垂直同期が高い方が見やすいことを忘れずに。

今回はかなり役に立つものがあるハズです。どれもつまみ調整が大変かもしれませんが,それなりの価値があるのではないでしょうか? 個人的には4番がおすすめです(映らないモニタも多いけど……)。

●CZ-6xx系(デュアルスキャン)

デュアルスキャンモニタは24kHzがなく,かなり臨界値も低いところにあります。垂

直同期も，31kHzモードでは，60Hzまでいかないので，インタレースなどはとても見られたものではありません。

また，ブラウン管自体も丸く，かなり見づらくなっています。さすがに安いモニタですから，これはかりはしかたがないかもしれませんね。

とりあえずこのシリーズ用にも対応するものをと，7番だけ作ってみました。確認はCZ-608でしかしていません。しかし，さすがに垂直同期が50Hzを切ると，かなり見づらくなりますねぇ。

CU21系，CZ-621系

CU21は，純正モニタのなかではかなり使いやすい部類に入ります。CZ-6xx系は普通，インタレースの広いモードとノンイン

表1　モニタ別CRTC設定表

●純正モニタCZ-6xx(3MODE SCAN)編

1)　ノンインタの広い1：1モード

水平ドット	840
垂直ドット	630
水平同期	31.273kHz
垂直同期	47.098Hz

レジスタ設定値
R00=$008A R01=$000F R02=$0017 R03=$0080
R04=$0297 R05=$0000 R06=$0021 R07=$0297
R20=$16　HRL=0

2)　ノンインタの広い3：4モード

水平ドット	560
垂直ドット	560
水平同期	30.500kHz
垂直同期	51.970Hz

レジスタ設定値
R00=$005E R01=$000D R02=$000C R03=$0052
R04=$024A R05=$0000 R06=$001A R07=$024A
R20=$15　HRL=0

3)　インタレースの広い1：1(に近い)モード

水平ドット	1024
垂直ドット	739
水平同期	25.870kHz
垂直同期	63.653Hz

レジスタ設定値
R00=$00A7 R01=$0009 R02=$001B R03=$009B
R04=$0195 R05=$0002 R06=$0024 R07=$0195
R20=$1A　HRL=0

4)　インタレースの広い1：1(に近い)モード

水平ドット	1024
垂直ドット	714
水平同期	25.870kHz
垂直同期	67.120 Hz(超臨界)

レジスタ設定値
R00=$00A7 R01=$0009 R02=$001B R03=$009B
R04=$0180 R05=$0000 R06=$001B R07=$0180
R20=$1A　HRL=0

5)　インタレースの広い3：4モード

水平ドット	960
垂直ドット	960
水平同期	30.612kHz
垂直同期	59.154 Hz

レジスタ設定値
R00=$008D R01=$0008 R02=$000D R03=$0085
R04=$0204 R05=$0002 R06=$0024 R07=$0204
R20=$1A　HRL=0

6)　オマケの640×480モード

水平ドット	640
垂直ドット	480
水平同期	31.500kHz
垂直同期	60.928Hz

レジスタ設定値
R00=$0089 R01=$000F R02=$0021 R03=$0071
R04=$0204 R05=$0002 R06=$0020 R07=$0200
R20=$16　HRL=0

●純正モニタCZ6xx(2 MODE SCAN)編

7)　ノンインタの広い1：1(に近い)モード

水平ドット	816
垂直ドット	578
水平同期	30.609kHz
垂直同期	49.777Hz

レジスタ設定値
R00=$008D R01=$000F R02=$001A R03=$0083
R04=$0266 R05=$0000 R06=$0023 R07=$0264
R20=$16　HRL=0

●純正モニタCU21，CZ-621編

8)　ノンインタの広い1：1(に近い)モード

水平ドット	824
垂直ドット	566
水平同期	31.500kHz
垂直同期	53.120Hz

レジスタ設定値
R00=$0089 R01=$0011 R02=$0019 R03=$0080
R04=$0250 R05=$0001 R06=$001A R07=$0250
R20=$16　HRL=0

9)　インタレースの広い1：1(に近い)モード

水平ドット	1024
垂直ドット	716
水平同期	26.344kHz
垂直同期	68.078 Hz

レジスタ設定値
R00=$00A4 R01=$0009 R02=$001A R03=$009A
R04=$0182 R05=$0001 R06=$001C R07=$0182
R20=$1A　HRL=0

10)　インタレースの広い1：1(に近い)モード

水平ドット	1000
垂直ドット	714
水平同期	27.686kHz
垂直同期	71.7309Hz

レジスタ設定値
R00=$009C R01=$0011 R02=$001B R03=$0098
R04=$0181 R05=$0001 R06=$001C R07=$0181
R20=$1A　HRL=0

11)　インタレースの広い1：1(に近い)モード

水平ドット	920
垂直ドット	714
水平同期	28.043kHz
垂直同期	72.043Hz

レジスタ設定値
R00=$009A R01=$000F R02=$001D R03=$0090
R04=$0181 R05=$0001 R06=$001C R07=$0181
R20=$1A　HRL=0

タレースのちょっとだけ広いモードとの間はつまみをいじらないと，絶対にいけませんが，CU21は同じつまみの位置で結構幅があり，なにもいじらずにいろいろできて便利です。

いちおうCZ-621系と書いてありますが，編集部ではCU21でしか試していないので，わかりません。でも多分大丈夫でしょう。

特徴といえば，このモニタは21インチでは格安であることです。値段なりの性能ではありますが，なかなかお買い得なモニタかもしれません。

なお，これはソフト屋さんにいいたいのですが，ゲームで特殊な画面モードを作るときに，垂直同期パルス幅を2line以下にすると，CU21では画面が縦に流れたまま止まらなくなってしまいます。具体的にいうなら，R05を1以下にしてはいけないということです。

データ表の7)～11)はかなりいろいろありますが，コントラスト以外のつまみをほとんどいじる必要がないというのが強いところです(かなり暗くなる)。特に11番はもうあっちの世界にいっています。発見した本人も，まさかここまでできるとは思ってもいませんでした。垂直同期約73Hz。いつでもできるわけではなく，これらの値がすべて揃っていないとうまくいきません。編集部のCU21では，レジスタの値をもはやひとつ臨界点に進めるだけで，映らなくなるという，正真正銘の臨界値です。10番，11番は2つとも，インタレースなのにかなり見やすいので，あなたのCU21でもできたら，喜んでください。

まとめ

これらのデータ設定は，来月号のオマケディスクに掲載予定のSADJUST.Rで設定できます。

もう，X680x0はドット比が……なんて，表面だけで判断するとぼけた意見に耳を貸すつもりはありませんし，X680x0は画面が狭いなどという意見にも耳を貸すつもりはありません（1024×1024できたら十分広いほうだと思う)。

CRTCの設定はかなりシビアで，動かしている本人がかなり熟知しないと，うまくいかない部分があります。多分，もう私が知っている限りのことは公開したハズです。

SX-WINDOWという空間を使いながら，ハードウェアの先まで使う喜び，アウトローかイリーガルか？　そんなことはともかく，マイコンしている喜びがあなたにも感じることができたら幸いです。

とりあえず，SX-WINDOWメガディスプレイ計画はこれで幕を閉じさせていただきたいと思います。

次にCRTCをいぢめるのはあなたです。

リスト1

```
  1: *$Author: Kohju $
  2: *$Header: a:/usr/home/Kohju/Labo/CRT/HighReso/highreso.has,v 1.1 1994/07/01 19:29:35 Kohju
Exp $
  3: *$Log: highreso.has,v $
  4: * Revision 1.1  1994/07/01  19:29:35  Kohju
  5: * Initial revision
  6: *
  7:
  8: *  ソースはCRTC960のソースを参考に作られています。みうチャン氏に感謝。
  9:
 10: *              インクルードファイルの設定        *
 11:
 12:                .include        DOSCALL.MAC
 13:                .include        IOCSCALL.MAC
 14:
 15: *              各種I／Oアドレスの設定           *
 16:
 17: TEXTVRAM       equ     $00e00000               * テキスト先頭アドレス
 18: CRTC           equ     $00e80000               * CRTCのアドレス
 19:
 20: *                      各種コードの設定          *
 21:
 22: READY          equ     $00                     * 正常終了コード
 23: ERROR          equ     $01                     * エラー終了コード
 24: TAB            equ     $09                     * タブ
 25: CR             equ     $0d                     * CR コード
 26: LF             equ     $0a                     * LF コード
 27: ESC            equ     $1b                     * エスケープ
 28: EOS            equ     $00                     * 終了コード
 29: STDERR         equ     $0002                   * 標準エラー出力
 30:
 31: *                      プログラム開始            *
 32:
 33: START:
 34:                bra     INIT
 35:
 36: *                      常駐部            *
 37:
 38: CRTMOD:
 39:                cmpi.b  #$0f,d1                 * $10以上は、特殊モードなので分岐
 40:                bhi     @f
 41:                btst    #0,d1                   * d1の最下位ビットが0ならば、31kHzモード
 42:                beq     @f                      * 15kHzモードはすべて対応した31kHzモードに設定し直す。
 43:                bclr    #0,d1
 44: @@:            cmpi.b  #$10,d1                 * CRTMODE 16を設定された。
 45:                beq     Change768x512
 46:                cmpi.b  #$11,d1                 * CRTMODE 17を設定された。
 47:                beq     Change1024x424
 48:                cmpi.b  #$12,d1                 * CRTMODE 18を設定された。
 49:                beq     Change1024x848
 50:                cmpi.b  #$13,d1                 * CRTMODE 19を設定された。
 51:                beq     Change640x480
 52:                cmpi.b  #$14,d1                 * CRTMODE 20を設定された。
 53:                beq     Change768x512
 54:                cmpi.b  #$15,d1                 * CRTMODE 21を設定された。
 55:                beq     Change1024x424
 56:                cmpi.b  #$16,d1                 * CRTMODE 22を設定された。
 57:                beq     Change1024x848
 58:                cmpi.b  #$17,d1                 * CRTMODE 23を設定された。
 59:                beq     Change640x480
 60:                cmpi.b  #$18,d1                 * CRTMODE 24を設定された。
 61:                beq     Change768x512
 62:                cmpi.b  #$19,d1                 * CRTMODE 25を設定された。
 63:                beq     Change1024x424
 64:                cmpi.b  #$1a,d1                 * CRTMODE 26を設定された。
 65:                beq     Change1024x848
 66:                cmpi.b  #$1b,d1                 * CRTMODE 27を設定された。
 67:                beq     Change640x480
 68:                lea.l   IOCS10(pc),a0           * もとの IOCS コールの格納アドレス
 69:                movea.l (a0),a0                 * もとの IOCS コールのアドレス
 70:                jmp     (a0)                    * 本来のルーチンへ
 71:
 72: Change768x512:
 73:                lea.l   IOCS10(pc),a0           * もとの IOCS コールの格納アドレス
 74:                movea.l (a0),a0                 * もとの IOCS コールのアドレスを取り出す
 75:                jsr     (a0)                    * ひとまず 768 x 512 ドットモードにしてお
 76: *              move.b    #$17,$E80029          * ドットクロックを100MHzにする。
 77: *○以上のコメントの説明。
 78: *  このコメントを外すと、50MHzオシレータを100MHzに変えたような改造マシンで、
 79: *768*512モードを選択した時、100MHzのオシレータでドットクロックを作成するので、
 80: *画面のちらつきが少なくなります。768*512モードでは動画、アニメはやらないと
 81: *思われたのでこうしてみました。
 82:                rts                             * リターン
 83:
 84: Change1024x4
24:
 85:                lea.l   IOCS10(pc),a0           * もとの IOCS コールの格納アドレス
 86:                movea.l (a0),a0                 * もとの IOCS コールのアドレスを取り出す
 87:                jsr     (a0)                    * ひとまず 1024 x 424 ドットモードにしておく
 88: *              move.w  #$00a4,$E80000          * 拡張モードに設定
 89: *              move.w  #$0004,$E80002          *         〃
 90: *              move.w  #$001d,$E80004          *         〃
 91: *              move.w  #$009d,$E80006          *         〃
 92: *              move.w  #$01b5,$E80008          *         〃
 93: *              move.w  #$0006,$E8000a          *         〃
 94: *              move.w  #$000d,$E8000c          *         〃
 95: *              move.w  #$01b5,$E8000e          *         〃
 96: *              move.b    #$17,$E80029          *         〃
 97: *○以上のコメントの説明。
 98: *  この1024*424モードもアニメーション、ゲームなどには使われないと思ったので、
 99: *できる限り垂直同期を上げて、見やすさを徹底してみました。
100: *  コメントのレジスタの値は、50MHz->100MHz改造マシンをターゲットに、垂直同期を、
101: *80Hz以上に上げているはずです。
102: *  自分のモニタにあった最良と思われる1024*424モードを作って下さい。
103:                rts                             * リターン
104:
105: Change1024x848:
106:                lea.l   IOCS10(pc),a0           * もとの IOCS コールの格納アドレス
107:                movea.l (a0),a0                 * もとの IOCS コールのアドレスを取り出す
108:                jsr     (a0)                    * ひとまず 1024 x 848 ドットモードにしておく
109: *              move.w  #$00a4,$E80000          * 拡張モードに設定
110: *              move.w  #$0004,$E80002          *         〃
111: *              move.w  #$001d,$E80004          *         〃
112: *              move.w  #$009d,$E80006          *         〃
113: *              move.w  #$01b5,$E80008          *         〃
114: *              move.w  #$0006,$E8000a          *         〃
115: *              move.w  #$000d,$E8000c          *         〃
116: *              move.w  #$01b5,$E8000e          *         〃
117: *              move.b    #$1b,$E80029          *         〃
118: *○以上のコメントの説明。
119: *  この1024*848モードもアニメーション、ゲームなどには使われないと思ったので、
120: *できる限り垂直同期を上げて、インタレスの見づらさを解消してみました。
121: *  コメントのレジスタの値は、50MHz->100MHz改造マシンをターゲットに、垂直同期を、
122: *80Hz以上に上げているので、インタレスでも見安いはずです。
123: *  μEMACSや、TeXのpreviewerなど1024*848モードに対応しているソフトが見違えて、
124: *使えるようになるはずです。
125: *  自分のモニタにあった最良と思われる1024*848モードを作って下さい。
126:                rts                             * リターン
127:
128: Change640x480:
129:                lea.l   IOCS10(pc),a0           * もとの IOCS コールの格納アドレス
130:                movea.l (a0),a0                 * もとの IOCS コールのアドレスを取り出す
131:                jsr     (a0)                    * ひとまず 640 x 480 ドットモードにしておく
132:                move.w  #$0089,$E80000          * 拡張モードに設定
133:                move.w  #$000e,$E80002          *         〃
134:                move.w  #$001c,$E80004          *         〃
135:                move.w  #$006c,$E80006          *         〃
136:                move.w  #$0237,$E80008          *         〃
137:                move.w  #$0005,$E8000a          *         〃
138:                move.w  #$0028,$E8000c          *         〃
139:                move.w  #$0208,$E8000e          *         〃
140:                move.b    #$16,$E80029          *         〃
141: *   この設定は普通のマシンで普通のモニタで再現出来ます。
142:                rts                             * リターン
143:
144: CHECK:
145:                .dc.b   'HiRs'                  * 常駐判定用
146:
147: MEMPOINTER:
148:                .dc.l   0                       * メモリ管理ポインタのバッファー
149:
150: IOCS10:
151:                .dc.l   0                       * 元の _CRTMOD のアドレス
152:
153: SCMODE:
154:                .dc.l   0                       * スクリーンモード
```

```
155:            .quad
156:
157: *                  常駐設定／解除ルーチン（非常駐部）              *
158: INIT:
159:            lea.l   MEMPOINTER(pc),a1           * メモリ管理ポインタのバッファのアドレス
160:            move.l  a0,(a1)                     * メモリ管理ポインタのバッファーに格納
161:            pea.l   TITLE_MES(pc)               * タイトルの先頭アドレス
162:            DOS     __PRINT                     * タイトルを表示
163:            addq.w  #4,sp                       * スタックを補正
164:            addq.w  #1,a2                       * コマンドラインの先頭
165:            bsr     SKIP_SPACE                  * タブ／スペースをスキップ
166:            move.b  (a2)+,d0                    * 1文字取り出す
167:            beq     KEEP
168:            cmpi.b  #'/',d0                     * スイッチがあるか?
169:            beq     SWITCH                      * その処理ルーチンへ
170:            cmpi.b  #'-',d0                     * スイッチがあるか?
171:            beq     SWITCH                      * その処理ルーチンへ
172:            bra     USAGE
173: KEEP:
174:            bsr     KEEPCHK                     * 常駐をチェック
175:            tst.w   d0                          * もう常駐しているか?
176:            beq     ERROR01                     * そうだったらエラー
177:            move.w  #$0110,-(sp)                * IOCS _CRTMOD
178:            DOS     __INTVCG                            * ベクタを得る
179:            addq.w  #2,sp                       * スタックを補正
180:            cmpi.l  #$fc0000,d0                 * ベクタはもう既に書き換えられているか?
181: *          blt     ERROR03                     * そうだったらエラー
182:            move.w  #$0110,d1                   * IOCS _CRTMOD
183:            lea.l   CRTMOD(pc),a1               * HighReso 処理先頭アドレス
184:            IOCS    __B_INTVCS                  * ベクタセット
185:            lea.l   IOCS10(pc),a0               * もとの IOCS コールの格納アドレス
186:            move.l  d0,(a0)                     * もとの IOCS コールのアドレスを格納
187:            pea.l   KEEP_MES(pc)                * 常駐メッセージの先頭アドレス
188:            DOS     __PRINT                     * 常駐メッセージを表示
189:            addq.w  #4,sp                       * スタックを補正
190:            clr.w   -(sp)                       * 終了コード
191:            move.l  #INIT-START,-(sp)           * 常駐部の長さ
192:            DOS     __KEEPPR                            * 常駐&終了
193: SWITCH:
194:            move.b  (a2)+,d0                    * 1文字取り出す
195:            andi.b  #%01011111,d0               * 大文字に変換
196:            cmpi.b  #'R',d0                     * 常駐解除したいのか?
197:            beq     RELEASE                     * その処理ルーチンへ
198:            bra     USAGE                       * 無効なスイッチのため使用法表示
199: RELEASE:
200:            bsr     KEEPCHK                     * 常駐をチェック
201:            tst.w   d0                          * 常駐していないか?
202:            bne     ERROR02                     * そうだったらエラー
203:            clr.l   -(sp)                       * スーパーバイザモードを指定
204:            DOS     __SUPER                     * スーパーバイザモードへ移行
205:            move.l  d0,(sp)                     * SSP を保存
206:            movea.l $440.w,a6                   * 現在の IOCS _CRTMOD のベクタ
207:            movea.l IOCS10-CRTMOD(a6),a1        * もとの IOCS コールのアドレスを取り出す
208:            move.w  #$0110,d1                   * IOCS _CRTMOD を指定
209:            IOCS    __B_INTVCS                  * 元のベクタをセット
210:            movea.l MEMPOINTER-CRTMOD(a6),a0 * 常駐しているプログラムのメモリ管理ポインタ
211:            DOS     __SUPER                     * ユーザーモードに復帰
212:            addq.w  #4,sp                       * スタック補正
213:            lea.l   16(a0),a0                   * 常駐しているプログラムの先頭を算出
214:            move.l  a0,-(sp)                    * 常駐しているプログラムの先頭を指定
215:            DOS     __MFREE                     * 常駐解除
216:            addq.w  #4,sp                       * スタック補正
217:            pea.l   KAIJYO_MES(pc)              * 常駐解除メッセージの先頭アドレス
218:            DOS     __PRINT                     * 常駐解除メッセージを表示
219:            addq.w  #4,sp                       * スタックを補正
220:            DOS     __EXIT                      * プログラム終了
221:
222: KEEPCHK:
223:            clr.l   -(sp)                       * スーパーバイザモードを指定
224:            DOS     __SUPER                     * スーパーバイザモードへ移行
225:            move.l  d0,(sp)                     * SSP を保存
226:            movea.l $440.w,a6                   * 現在の IOCS _CRTMOD のベクタ
227:            cmpi.l  #'HiRs',CHECK-CRTMOD(a6) * 常駐しているか?
228:            bne     NO_KEEP_HighReso                    * していなかったらその処理へ
229:            DOS     __SUPER                     * ユーザーモードに復帰
230:            addq.w  #4,sp                       * スタック補正
231:            moveq.l #0,d0                       * 常駐している
232:            rts                                 * リターン
233: NO_KEEP_HighReso:
234:            DOS     __SUPER                     * ユーザーモードに復帰
235:            addq.w  #4,sp                       * スタック補正
236:            moveq.l #$ff,d0                     * 常駐していない
237:            rts                                 * リターン
238:
239: SKIP_SPACE:
240:            cmpi.b  #' ',(a2)                   * スペースがあるか?
241:            beq     SKIP                        * その処理ルーチンへ
242:            cmpi.b  #TAB,(a2)                   * タブがあるか?
243:            beq     SKIP                        * その処理ルーチンへ
244:            rts                                 * リターン
245: SKIP:
246:            addq.w  #1,a2                       * 1文字スキップ
247:            bra     SKIP_SPACE                  * ループ
248:
249: USAGE:
250:            lea.l   USAGE_MES(pc),a1            * エラーメッセージの先頭アドレス
251:            bra     ERREND                      * エラー終了処理へ
252: ERROR01:
253:            lea.l   ERR_MES1(pc),a1             * エラーメッセージの先頭アドレス
254:            bra     ERREND                      * エラー終了処理へ
255: ERROR02:
256:            lea.l   ERR_MES2(pc),a1             * エラーメッセージの先頭アドレス
257:            bra     ERREND                      * エラー終了処理へ
258: ERROR03:
259:            lea.l   ERR_MES3(pc),a1             * エラーメッセージの先頭アドレス
260: ERREND:
261:            move.w  #STDERR,-(sp)               * 標準エラー出力
262:            move.l  a1,-(sp)                    * エラーメッセージのアドレス
263:            DOS     __FPUTS                     * 表示
264:            addq.w  #6,sp                       * スタック補正
265:            move.w  #ERROR,-(sp)                * エラー終了コード
266:            DOS     __EXIT2                     * エラー終了
267:
268:
269: *                  データ領域                                      *
270:
271: TITLE_MES:
272:            .dc.b   'Human68K HighReso Ver 1.00 Copyleft(C)1994 Kohju/Taki',CR,LF
273:            .dc.b   '$Id: highreso.has,v 1.1 1994/07/01 19:29:35 Kohju Exp $',CR,LF,EOS
274: USAGE_MES:
275:            .dc.b   '機能：ハイレゾモニタ用に、30kHz以下のモードをすべて30kHz以上に持ち上げます。',CR,LF
276:            .dc.b   '  また24kHzモードは、超高解像度にします。',CR,LF,CR,LF
277:            .dc.b   'オプション：',CR,LF
278:            .dc.b   '  -R   常駐解除します。',CR,LF
279:            .dc.b   '',CR,LF
280:
281: KAIJYO_MES:
282:            .dc.b   'HighReso の常駐を解除しました。',CR,LF,EOS
283:
284: ERR_MES1:
285:            .dc.b   'HighReso は既に常駐しています。',CR,LF,EOS
286:
287: ERR_MES2:
288:            .dc.b   'HighReso は常駐していません。',CR,LF,EOS
289:
290: ERR_MES3:
291:            .dc.b   'IOCS _CRTMOD のベクタが書き換えられています。',CR,LF
292:            .dc.b   '常駐できません。',CR,LF,EOS
293:
294: KEEP_MES:
295:            .dc.b   'HighReso は常駐しました。',CR,LF,EOS
296:
297:            .end
```

リスト2

（757バイトでセーブ）

```
000000  23 E5 2D 6C 68 35 2D CF : 3A
000008  02 00 00 FE 03 00 00 93 : 96
000010  2A FE 1C 20 01 0A 68 69 : 40
000018  67 68 72 65 73 6F 2E 78 : 2E
000020  94 D3 48 00 00 02 8B 6B : A7
000028  9D F6 AD 27 2D FF 6E AB : AC
000030  29 E9 68 DC 8A 2B 56 13 : 74
000038  E8 42 68 B2 29 69 68 0A : 48
000040  58 40 B4 F0 93 D6 9B 7B : BB
000048  DA 79 31 09 11 0B 48 87 : 78
000050  D4 01 A5 80 51 AD E0 25 : FD
000058  F7 DF 70 25 15 6A 02 C7 : B3
000060  84 78 77 06 D2 67 4D 37 : 36
000068  06 31 B3 23 B7 0B A4 49 : BC
000070  06 43 81 91 78 0E 25 64 : 6A
000078  48 C8 B2 AF 1B 65 8F BF : 3F
-----------------------------------
CKSUM:  CD 8C D7 AB E5 20 E4 07 748D

000080  FA A2 00 F0 BF 81 1A B0 : 96
000088  05 C6 EE 4B F0 0D BF BE : 7E
000090  AF F0 3F FF A1 80 79 9B : 12
000098  32 0D E5 F2 0D 59 00 15 : 91
0000A0  A3 55 80 06 77 05 84 27 : A5
0000A8  EB 2A 62 77 A5 54 4D 31 : 65
0000B0  6E 44 EF 8A F0 4E FC B5 : 1A
0000B8  E2 69 CA F8 4F 00 B6 42 : 54
0000C0  78 25 72 27 84 5A 81 2E : C3
0000C8  4B 52 27 86 5F B2 04 F0 : 4F
0000D0  CA 27 DE 9E A8 BC 91 7D : DF
0000D8  84 E4 0C 38 9B 30 DF 6C : C2
0000E0  C3 74 C4 2F 68 04 C0 D7 : 2D
0000E8  00 17 4A 2D 98 BE 20 B6 : BA
0000F0  82 F1 05 EE DE D0 67 C9 : 44
0000F8  16 D4 67 DD 0B 40 5F 14 : EC
-----------------------------------
CKSUM:  2A 63 AA D5 C7 D8 70 DE 48BB
```

```
000100  5B 61 E7 6A 2E 96 9A 80 : EB
000108  53 8B E3 27 21 2C 15 71 : BB
000110  F7 3D 8D 19 EB FB AA 72 : DC
000118  D2 94 14 17 A4 BA 51 58 : 98
000120  51 00 3B CE 1C 95 C3 60 : 2E
000128  07 45 B5 13 C8 2D 90 FE : 97
000130  0F FD 44 CA 84 FC A0 1F : 59
000138  CD 9D 8B 27 0B D7 5C 28 : 82
000140  DA 58 50 00 78 32 A3 3F : 0E
000148  E1 AD 1E 5A 6A 40 9E BF : 0D
000150  EA 49 69 4A 67 86 C6 26 : BF
000158  D0 AE AC 47 3D E8 5E B1 : A5
000160  C3 96 60 17 A3 89 54 AC : FC
000168  E1 C6 89 5F 09 FE 06 45 : E1
000170  A7 E6 B9 2A E5 F1 E4 B4 : DE
000178  7F 53 18 3C AC 61 58 C0 : 4B
-----------------------------------
CKSUM:  EA 2D 67 5A 14 C5 F4 9A 831D

000180  92 30 78 D7 25 D2 90 35 : CD
000188  C1 C3 A6 AB C1 BA 51 29 : CA
000190  4C BA 2C 63 57 0D AB 16 : BA
000198  DF B5 1F 43 D5 C0 6A C5 : BA
0001A0  5C 70 4E 45 16 BA 72 1B : BC
0001A8  6F 40 85 6A 26 90 AC D3 : D3
0001B0  90 AC 28 7E C4 67 87 6A : FE
0001B8  18 F9 0B 2E 64 34 04 E2 : C8
0001C0  C3 68 26 65 9D 88 33 75 : 83
0001C8  48 BD 45 DA 83 0B 25 FE : D5
0001D0  D4 A4 74 22 46 AF E5 5B : 43
0001D8  91 2C 12 C5 56 FF 1F 14 : 1C
0001E0  97 37 F8 C4 AC AB 76 EC : 43
0001E8  88 E2 C7 E9 62 5F E0 64 : 1F
0001F0  71 51 E3 56 58 58 56 92 : 93
0001F8  DF 16 2E 14 87 57 11 30 : 56
-----------------------------------
CKSUM:  D0 2C 30 C0 1F 38 B8 67 C29F
```

```
000200  F0 5B A1 55 07 9E F8 91 : 6F
000208  46 CB 18 6C AA E2 C4 C7 : AC
000210  E3 E4 8D A5 61 18 A8 74 : 8E
000218  EF 92 E9 DB 11 3E 79 60 : 6D
000220  F9 ED 77 C6 DC 9C B9 31 : 85
000228  C9 54 DD 00 EA 3F EC DD : EC
000230  AD 6C 78 C7 44 FD 53 F3 : DF
000238  8E 63 E1 1E F3 B7 1C DA : 90
000240  6B 40 7A EF 0D 29 59 7E : 21
000248  19 9D E3 6A 1C CB 73 4F : AC
000250  14 D3 8D A4 6F 21 BC 1B : 7F
000258  1D 13 7C E6 D3 EB F7 0D : 54
000260  FE 8B FD CD AC 6D 25 AD : 3E
000268  1B A0 B3 F1 87 BE F2 B4 : 4A
000270  68 36 2C FC 4B 40 CE FE : 1D
000278  3A C6 99 3B 37 43 3E 49 : D5
-----------------------------------
CKSUM:  75 96 B7 C4 40 13 93 A4 90EE

000280  B4 CD A3 F2 AA 6E 81 E0 : 8F
000288  9F 24 F7 27 E9 9F 3D 6B : 11
000290  66 E8 10 9C 85 69 26 FA : 08
000298  F3 B3 E6 4E 9B A1 B9 4D : 1C
0002A0  B3 F8 49 B5 18 EE 34 ED : D0
0002A8  9A 0D 13 77 D8 A2 0D AA : 62
0002B0  1F 16 67 BC DA 74 69 74 : 83
0002B8  4D E0 37 E3 F2 C2 6E 50 : B9
0002C0  FC FB 7F B3 7B 68 F7 C1 : C4
0002C8  51 1E 61 2F 51 56 E3 EE : 77
0002D0  52 D4 74 E7 92 74 CF 78 : CE
0002D8  6F D2 6E 69 BB B9 9A 46 : 6C
0002E0  FC 8D E9 37 AB 73 1F 47 : 2D
0002E8  07 FF 1B BB B7 81 FA 68 : 76
0002F0  76 BC 7C 00 00 00 00 00 : AE
0002F8  00 00 00 00 00 00 00 00 : 00
-----------------------------------
CKSUM:  EC 8E CC F2 EA BC 11 09 5CDB
```

ごめんなさいのコーナー(特別版)

8月号のSX-PICSLICEの記事でリスト1(53ページ)が完全には掲載されておらず,ご迷惑をおかけしました。以下に再掲載いたします。

```
436: /*************************************************************************
437:  *       dropIcon():      アイコンのドロップ処理
438:  *************************************************************************
439:  */
440: void
441: dropIcon(void )
442: {
443:         int errCode, len;
444:         Rect rc;
445:         Drag *dragPtr;                          /* ドラッグポインタ          */
446:         Cell *pcell;                            /* セルレコードへのポインタ     */
447:         IcState *pis;                           /* アイコン管理レコードへのポインタ */
448:         char fileName[TS_NAMEMAX];              /* ファイル名                */
449:
450:         /* ドラッグポインタを取得する */
451:         errCode = TSGetDrag(&dragPtr);
452:         if (errCode < 0)
453:                 /* ドラッグレコードが無い */
454:                 return;
455:
456:         /* セルリストハンドルをロックする */
457:         MMHdlLock(dragPtr->cellListHdl);
458:         /* セルレコードへのポインタを取得する */
459:         pcell = *dragPtr->cellListHdl;
460:         /* ラバーバンドを消去する */
461:         TSHideDrag();
462:         /* セルレコードの情報の種類がアイコン管理レコード(上位ワードが'FS')か? */
463:         if (HIWORD(pcell->kind) == 'FS') {
464:                 /* アイコンは1つ(ドラッグレコードのセルリストの長さがアイコン
465:                  * 管理レコードを含むセルレコードの長さと同一の場合)か?
466:                  */
467:                 if (dragPtr->length == sizeof(IcState) + 8 ) {
468:                         /* アイコン管理レコードへのポインタを取得する */
469:                         pis = (IcState *) pcell->data;
470:                         /* ファイルの属性をチェック */
471:                         if ((pis->attr & ATTRMASK) == TS_ARCH) {
472:                                 /* アイコンのフルパスを取得する */
473:                                 len = TSISRecToStr(
474:                                         pis, /* アイコン管理レコード     */
475:                                         fileName); /* パス名格納ポインタ */
476:                                 if (len != 0) {
477:                                         /* ドラッグを終了する */
478:                                         TSEndDrag(TS_FINISH);
479:                                         /* ドラッグされたファイルの読み込み */
480:                                         loadFile(fileName);
481:                                 }
482:                         } else          /* ファイルアイコン以外の場合    */
483:                                 /* アイコンを元の位置までドラッグして終了する */
484:                                 TSEndDrag(TS_PUTBACK);
485:                 } else
486:                         /* アイコン2つ以上か他のドラッグレコードだったら */
487:                         /* アイコンを元の位置までドラッグして終了する */
488:                         TSEndDrag(TS_PUTBACK);
489:         }
490:         /* セルリストハンドルのロックを解除する */
491:         MMHdlUnlock(dragPtr->cellListHdl);
492:         /* ドラッグされたファイルの内容を描画する */
493:         rc = windowPtr->graph.rect; /* ウィンドウ内全体を書き換える */
494:         GMSlideRect(&rc, GMLocalToGlobal(0)); /* グローバル座標系に変換する */
495:         WMAddRect(&rc);                         /* アップデートリージョンに追加する */
496: }
497:
498: /*************************************************************************
499:  *       showErrDialog():        エラーダイアログの表示
500:  *************************************************************************
501:  */
502: void
503: showErrDialog(void )
504: {
505:         int i;
506:
507:         static struct {                         /* エラーメッセージ      */
508:                 int code;                       /* エラーコード          */
509:                 int manager;                    /* 使用するマネージャ    */
510:                 int flag;                       /* フラグ情報            */
511:                 char *str;                      /* 表示する文字列        */
512:         } errorMsg[] = {
513:                 {  1, 1, 1, "SX SYSTEMのバージョンが違います。" },
514:                 {  2, 2, 1, "ウィンドウが作成できません。" },
515:                 { 11, 1, 1, "ファイルがオープンできません。" },
516:                 {  0, 1, 1, "エラーが発生しました。" }
517:         };
518:
519:         for (i = 0; errorMsg[i].code != 0; i++) /* エラーコードを捜す   */
520:                 if (errorMsg[i].code == errorCode)
521:                         break;
522:         switch (errorMsg[i].manager) {  /* 利用するマネージャの選択      */
523:         case 1:                                 /* ダイアログマン            */
524:                 DMError(errorMsg[i].flag, errorMsg[i].str);
525:                 break;
526:         case 2:                                 /* タスクマン                */
527:                 TSErrDialogN(errorMsg[i].flag, errorMsg[i].str);
528:                 break;
529:         }
530: }
531:
532: /*************************************************************************
533:  *       endProc():      終了手続き
534:  *************************************************************************
535:  *       引数:   int code        終了コード
536:  *       注釈:   ハンドルの解放やウィンドウの廃棄と、プログラムの終了を行う。
537:  */
538: void
539: endProc(int code )
540: {
541:         /* ウィンドウポインタが確保されたままか? */
542:         if (windowPtr != NULL)
543:                 WMDispose(windowPtr); /* ウィンドウを廃棄する   */
544:
545:         MMHdlDispose(dataHdl);
546:
547:         /* ファイルが開かれたままか? */
548:         if(drawing)
549:                 TSClose(psi.filehandle);
550:
551:         exit(code);                             /* プログラムを終了する        */
552: }
553:
554: /*************************************************************************
555:  *       loadFile():     ファイル読み込み
556:  *************************************************************************
557:  *               char *pname     ファイル名へのポインタ
558:  *       戻り値: BOOLEAN         = TRUE:  読み込み成功
559:  *                                = FALSE: 読み込み失敗
560:  */
561: BOOLEAN
562: loadFile(char *pname)
563: {
564:
565:         int             fn;
566:         int             ret;
567:
568:         /* ファイルが開かれたままか? */
569:         if(drawing) {
570:                 MMPtrDispose( psi.filebuf );
571:                 MMPtrDispose( psi.chainbuf );
572:                 TSClose(psi.filehandle);
573:         }
574:
575:         /* 新たにファイルを開く */
576:     if ( (fn = TSOpen( pname, 0 )) < 0 ) {
577:         errorCode = 11; /* オープンできなかった          */
578:                 return FALSE;           /* 失敗したのでFALSEを返す      */
579:         }
580:
581:         /* あらかじめ設定しておくpicsliceinfoのメンバ */
582:         psi.filehandle = fn;
583:         psi.filebuf = (unsigned char *)MMChPtrNew( FILEBUFSIZE );
584:         psi.filebufsize = FILEBUFSIZE;
585:         psi.chainbuf = (unsigned short *)MMChPtrNew( CHAINBUFSIZE );
586:         psi.chainbufsize = CHAINBUFSIZE/sizeof(unsigned short);
587:         /* picsliceローディングの開始 */
588:         ret = picsliceOpen( &psi, PICSLICE_GRAM );
589:
590:         strcpy(filename, pname);
591:         drawGraph();
592:         drawing = TRUE;
593:         y = 0;
594:         return TRUE;                            /* 成功したのでTRUEを返す       */
595: }
596:
597: /*********************************/
598: /*              文字列を送信する                */
599: /*********************************/
600: void
601: SendMes(int id, char *fmt, ...)
602: {
603:         char    **hdl, *p, *q, *sval;
604:         int             i, ret;
605:         double  f;
606:         int     cnt = 10;       /*メッセージのやりとりを10回行なって
607:                                         それでもダメならエラーにする      */
608:         TsEvent eventRec;
609:         va_list ap;     /* 各名なし引数を順々に渡す */
610:
611:         va_start(ap, fmt);
612:         hdl = (char **)MMChHdlNew(1024);
613:         if(hdl == NULL) {
614:                 TSErrDialogN(0x101,"メモリが確保出来ません");
615:                 return;
616:         }
617:         q = *hdl;
618:         for(p = fmt; *p; p++) {
619:                 if(*p != '%') {
620:                         *q++ = *p;
621:                         continue;
622:                 }
623:                 switch(*++p) {
624:                         case 'd':
625:                                 i = va_arg(ap, int);
626:                                 ret = sprintf(q, "%d", i);
627:                                 q = &q[ret];
628:                                 break;
629:                         case 'f':
630:                                 f = va_arg(ap, double);
631:                                 ret = sprintf(q, "%f", f);
632:                                 q = &q[ret];
633:                                 break;
634:                         case 's':
635:                                 sval = va_arg(ap, char *);
636:                                 while(*sval)    *q++ = *sval++;
637:                                 break;
638:                         default:
639:                                 *q++ = *p;
640:                                 break;
641:                 }
642:         }
643:         va_end(ap);     /* クリーンアップ */
644:         *q = '¥0';
645:
646:         eventRec.ts.whom   = (long)hdl;
647:         eventRec.ts.when   = EMSysTime();
648:         eventRec.ts.what2  = SX_BASIC_SEND;
649:         do {
650:                 ret = TSSendMes(id, &eventRec);
651:                 cnt--;
652:         } while(ret != 0 && ret != 2 && cnt != 0);
653:
654:         MMHdlDispose(hdl);
655:         if(cnt == 0) {
656:                 TSErrDialogN(1, "無効なタスクに通信を行ないました");
657:         }
658: }
```

(で)のショートプロぱーてぃ──その60

スクリーンセーバーで燃え燃え!

Komura Satoshi 古村 聡

今月でショートプロぱーてぃも60回,きっちり5年分です。約5年の歳月を経て,(で)氏の成長はいかがなものでしょうか。今月のプログラムはアプリケーションが2本にゲームが1本。たまにはプログラミングでのんびりとお楽しみあれ。

illustration : T.Takahashi

最近のスクリーンセーバーってすごいですよね,派手で。私,原稿の打ち込みは実をいうとX68000でない(ごめんなさい)ほかのマシンのウィンドウシステムでやってるんですが,スクリーンセーバーを手に入れました。最近の流行にもれず,そのマシンでは,240×150ドットくらいの大きさで取り込み画像がちまちま動くシステムがあるんですが,このスクリーンセーバーではその動画を再生してくれるのです。しかも,その動画はサンプルだけでなく,自分でビデオなどから取り込んで好きなデータを作れるっていうものだもんだからデータ作りに燃えた燃えた。最新式のビデオキャプチャーボードまで買ってガンガン取り込みました。おかげでうちのマシンはちょっと手を休めると亜美ちゃんのシャボンスプレーで燃え〜の,チャチャのパンダ体操で燃え〜の,スクルドのトンカチで燃え〜の,540Mバイトのハードディスクのうち400Mバイト以上を動画データで埋め尽くすアニメ上映燃え燃えマシンになってしまったんであります。

しかし,あんまり燃え燃えなシーンばかりがしかもたでつづけにランダムに上映されるので,ちょっと手を放すと目がアニメ(アッチのせかい)にくぎづけになって二度と作業に復帰できない困ったシステムになってしまったのでありました。うー,だってチャチャが,チャチャがかあいいほど愛くるしいだよほ〜っ!

うー,スクリーンセーバーがあると原稿があがら〜んっ! すでに日本語が変だし。え? つまらずにすらすら書けばスクリーンセーバーは動かんだろうって? ……そんなぁ。

清く正しく美しく!

てなわけでX68000のほうは原点に返ったスクリーンセーバーの登場です。コンソール画面上で使えるHuman68k用スクリーンセーバーSSAVER.Xです。どうぞっ。

SSAVER.X for X680x0

(要アセンブラ,リンカ)

兵庫県 米田和久

スクリーンセーバーといえば,それこそいまではアニメが上映されたり,トースターが飛んだり,魚が泳いだりと賑やかです。でも,もともとはスクリーンセーバーって画面の焼きつきをふせぐために,画面が一定時間同じだったら画面を消すというのが本来のお仕事だったのですよね。そう,このプログラムはキーボードなどからの入力がなかったらただ画面を暗くするだけの,清く正しく,原点に返ったものなのであります。

まず,ED.Xなどのエディタでリストを注意深く入力し,SSAVER.Sという名前でファイルをセーブして終了します。それから,AS.XやHAS.Xなどのアセンブラでアセンブルし,LK.X,HLK.Xなどのリンカでリンク作業をしてSSAVER.Xという実行ファイルを作ってください。

プログラムを実行すると,スクリーンセーバーはメモリに常駐し,X68000に一定時間キーボードやマウスなどの入力がなければ画面のコントラストを下げます。

実行するにはコマンドライン上で,

A>SSAVER.X

で常駐します。この場合,デフォルト動作となり10分間入力がなければコントラストを下げます。せっかくですから,AUTOEXEC.BATに入れておくといいでしょう。常駐を解除するには,もう一度コマンドラインから,実行すればOKです。

それから,このプログラムにはいろいろなスイッチも用意されています。

まず,/rで常駐解除。/tでコントラストを下げる待ち時間を設定できます。スイッチの直後に時間(単位は分で設定範囲は1〜30分)を書いてください。たとえば,

A>SSAVER /t5

で5分後に暗くなります。

/cは,コントラストの度合いを設定します。0で真っ暗,1で普通に暗く,2では薄暗くなります。/tと同じようにスイッチの直後に数字を書いてください。

/hを指定すると簡単なヘルプメッセージが表示されます。このオプションが指定されたときはスクリーンセーバーの常駐/解除は行われません。

これぞまさしく清く正しい,純正スクリーンセーバー! 基本的なスクリーンセーバーってこんなに簡単にできるものだったんですね。びっくりです。ショートにしてはちょっと長いけど。これを改良していろいろなモジュールを合わせていくスクリーンセーバーカーネルとかにできないもんでしょうか。誰か挑戦してみません? できたら,モジュールで「一定時間入力がないとハードディスクの内容をMOにバックアップするモジュール」とか……結構いいアイデアだと思いませんか。ちょっと怖い気もするけど。

そうそう,このプログラムでは入力を監視するデバイスはキーボード,マウス,ジョイステックになっています。だからキー

ボードもマウスも使わないようなプログラム，たとえばゲームのときにも画面が暗くなったりしません。よく某国民機のDOS版のスクリーンセーバーだとマウスを監視していなくて不便なことがありましたけど……，偉い。そうそう，時間経過を計るためのタイマーにはRTC16Hzパルスを使用しています。ほかのプログラムとぶつかることはまずないと思いますけど，使うときには気をつけましょうね。

あ～，それにしてもシンプルなスクリーンセーバーっていいですね。原稿書きにはX68000がいちばんかもしれないですね。うん。ああっ，しかし，X68000ばっかりさわっていると隣のマシンのスクリーンセーバーが動き始めちゃったりするんだけど。チャチャ燃え～。

SX-WINDOWも負けてない！

さて，2本目のプログラムにいきましょう。このプログラムはSX-WINDOWを使っている人のためのとっても便利なプログラムです。どうぞっ。

OBSCURE.X for SX-WINDOW

（要アセンブラ，リンカ，

SX-WINDOW ver.2.0以降）

東京都　鎌田　誠

SX-WINDOWで シャーペン.X などを使ってプログラムや文章を書いているとき，マウスポインタが邪魔だと思ったことはありませんか。あの白い三角のマウスポインタ，使わないときは消してしまいましょう。このプログラムは，アクティブウィンドウ上にマウスカーソルがあるときに邪魔なマウスポインタを消してくれます。

このプログラムを使うには，SX-WINDOWを用意するのは当然として，それと，アセンブラ，リンカが必要になります。

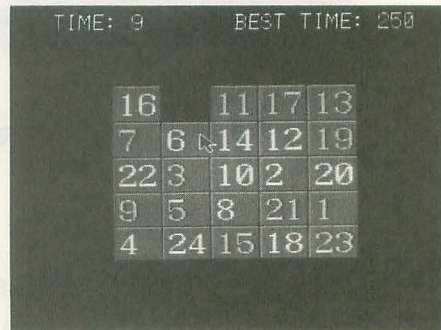
Human68k上のエディタED.XやSX-WINDOWのエディタなどでリスト2を打ち込みOBSCURE.Sという名前でセーブしてください。それから，アセンブル，リンク作業をして実行ファイルをOBSCURE.Xという名前で作ります。リンクする際にも特別なライブラリなどは必要ありません。

さて，あとはダブルクリックで実行してもいいですが，できればSX-WINDOWが動いているときに，常にこのOBSCUREが働いてくれるように，スタートアップメンテに登録してみてください。パラメータはいりません。そしてSX-WINDOWを再起動してみましょう。システムメニューで「再起動」を選択して，と。

アクティブウィンドウ上にマウスカーソルをおいて，キーボードからなにか文字を入力してみてください。ほら，マウスカーソルが消えたでしょう。マウスを動かしたり，ボタンを押したりすることでもう一度カーソルを表示できます。

OBSCUREがすでに組み込まれている状態でもう一度OBSCUREを実行すると，機能についての説明と「確認」「解除」と書かれたボタンのついたダイアログが表示されます。このダイアログが表示されたときに「解除」ボタンを押すとOBSCUREを解除することができます。解除しない場合は「確認」ボタンを押してください。

これって，あるあるある！って感じですよね。誰もが感じていたことをさらっと解決するんだから，いいですよね。私も，ほかのマシンのウィンドウでこれを書いてるんですが，あるといいですよね，こういう機能って。やっぱりSX-WINDOWで書くべきなのかな，うん。

そうそう，作者の鎌田さんはこのOBSCUREをフリーウェアにするそうです。せっかくの便利なプログラムですから，パソコン通信や手渡しでどんどん広めましょう！

PANEL.BAS

わしも自分のネットにアップロードしとこっと。

最後はゲームなのだ！

これまでの2本のプログラムは，アセンブラのリストで「打ち込めないよ～」と困った人もいたかもしれないですね。でもご安心。最後のプログラムはX-BASIC用のゲームプログラムです。どうぞっ。

PANEL.BAS for X680x0

（X-BASIC，要XVI.FNC）

千葉県　森川忠敬

15パズルって知ってますか？　そう，4×4の16個のマスの中にランダムに1～15までの数字の書いたパネルと空白が1個あって，パネルを動かして1から15まで順番に並べるというあのパズルゲームですね。このPANEL.BASは5×5＝25個のマスにしたパズルゲームなのです。

リスト3はX-BASIC用ですので，打ち込んでRUNすればOK……といきたいのですが，このプログラム，リストのままだと本誌1992年11月号に掲載されたX-BASIC用外部関数のXVI.FNCが必要です。ですから，X-BASICのBASIC.CNFに，

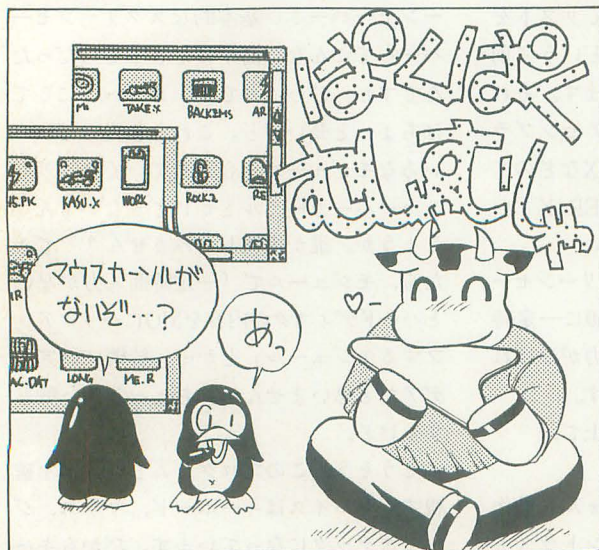
```
FUNC = XVI
```

としてXVI.FNCが使えるようにしておいてください。XVI.FNCがない場合には，

```
VWAIT(0)
```

と書いてある部分を削ってください。パネルを動かすスピードに関する部分ですが，ゲーム内容には関係ないのでXVI.FNCがなくても大丈夫でしょう。わかったかな？

さて，無事にリストを打ち終わったらセーブして，次は遊び方。

このゲームはマウスで操作します。動かしたいパネルの上下左右のいずれかの方向

に空白があるときにそのパネルを左クリックすると，空白の位置に移動します。でもって，バラバラになっている数字を左上から1から順番に並べ替えるだけです。パネルはまとめて移動させることもできます。空白から2つとか3つとか先のパネルを左クリックすれば，パネルはまとめて移動します。ギブアップしたい場合にはマウスの右ボタンをクリックしてください。

コンパイルして使うときには，

A>CC PANEL.BAS XVI.O

と，XVI.FNCのオブジェクトも忘れずにコンパイルしてくださいね。

このプログラム，とてもきれいに書けていて，拡張性に富んだいいリストだと思います。たとえば，80行にある“num”という変数に代入されている数を変更するとパネルの縦横を変更することができるんですよね。ちなみに指定できる数は2～5で，それ以外を指定するとちゃんと5になってくれます。う～ん，親切設計。

そうそう，プログラムをコンパイルして実行する場合には，

A>PANEL.X 4

などとしてパネルの縦横を変更できます。

そう，先月号で紹介したb_argc，b_argvを使ってプログラムに引数を渡しているんですね。同じプログラムを書くのでも，これだけ親切にできているのはすごいと思います。偉い。

しかし，パソコンってーのは本当にいろいろできるもんですね。スクリーンセーバーを作り動画を取り込んだり，カーソルを消したり，パズルを作ってしまったり。

しかも，この楽しみが自分の気に入るように作れるわけで，作られた時点でなにに使うかが決まってしまっているゲーム専用機とも，ワープロ専用機とも違うわけです。そう考えてみるとプログラミングっていうのはいちばんパソコンらしさを味わえる部分なのでありますね。そりゃ，スクリーンセーバー用の動画ファイルを作るのもいいですけどね。それじゃ，なんかプログラム作ろうかな。スクリーンセーバーのモジュールのネタでも。ん……。いかん，手を休めて考えていると，また“アニメ動画再生燃え燃え”が始まってしまう。そんな幸せをかみしめつつ，では，また来月。

リスト1 SSAVER.S

```
  1: *居座りぷろぐらむ
  2: *
  3: *                 SCREEN SAVER Ver1.1
  4: *
  5: *キーボード・マウス・ジョイスティク入力無=スクリーンコントラストダウン
  6: *
  7: *                              by K.yoneda '93.5.16
  8: *
  9:         .include    iocscall.mac
 10:         .include    doscall.mac
 11:         .text
 12:         .even
 13: *--------------------->> 常駐部 <<--------------------
 14: idn:    dc.b    'ScSav1.1'
 15: vct:    dc.l    0
 16:
 17: inter:  movem.l d0-d1/a0-a1,-(sp)
 18:         lea.l   work,a1         *work pointer a1
 19:         move.l  #$e8e001,a0     *システムポート#1
 20:         move.b  $e8802f,d1      *キーボード check
 21:         rol.b   #1,d1
 22:         bcc     quit
 23:         IOCS    _MS_GETDT       *マウス check
 24:         move.b  $e9a003,d1      *ジョイスティック #2 check
 25:         and.b   $e9a001,d1      *    ''      #1 check
 26:         not.b   d1
 27:         or.b    d1,d0
 28:         tst.l   d0              *マウス & ジョイスティック 入力の有無
 29:         bne     quit
 30:         addq.w  #1,(a1)         *count
 31:         move.w  (a1),d0
 32:         cmp.w   2(a1),d0        *待時間
 33:         bne     quit2
 34:         clr.w   (a1)            *count reset "0"
 35:         clr.b   4(a1)           *flag  reset "0"
 36:         move.b  5(a1),(a0)      *contrast down
 37:         bra     quit2
 38: quit:   clr.w   (a1)
 39:         tst.b   4(a1)           *flag check
 40:         bne     quit2
 41:         move.b  #$0e,(a0)       *Human68kデフォルト
 42:         addq.b  #1,4(a1)
 43: quit2:  movem.l (sp)+,d0-d1/a0-a1
 44:         rte
 45: work:
 46: count:  dc.w    0
 47: wtime:  dc.w    9600            *10分の場合のカウント
 48: flag:   dc.b    0
 49: ctras:  dc.b    5               *"暗い"に設定
 50: *---------------------------------> ここまで常駐 ---
 51:         .even
 52: start:  clr.l   a1              *始まり始まり
 53:         IOCS    _B_SUPER
 54:         move.l  d0,spsave
 55:         clr.w   d6
 56: resta:  movea.l (a0),a0         *分身の有無
 57:         cmpa.l  #$10000,a0
 58:         bcs     unknown
 59:         tst.l   (a0)
 60:         beq     unknown
 61:         lea.l   $100(a0),a6
 62:         move.l  (a6),d1
 63:         cmpi.l  #'ScSa',d1      *注意して入力 ScSa
 64:         bne     resta
 65:         move.l  4(a6),d1
 66:         cmpi.l  #'v1.1',d1      *注意して入力 v1.1
 67:         bne     resta
 68:
 69:         move.b  d1,d6
 70:         bsr     checksw
 71:         ori.w   #$0700,sr       *割り込み禁止
 72:
 73:         clr.b   d1              *RTC割り込み禁止
 74:         or.b    d1,$e88009      *
 75:         or.b    d1,$e88015      *
 76:         move.b  #%0000_1100,d1  *RTCタイマー OFF
 77:         move.b  d1,$e8a01f      *
 78:         movea.l a6,a1
 79:         addq.l  #8,a1
 80:         movea.l (a1),a1         *ベクターを元に戻す
 81:         move.w  #$40,d1         *
 82:         IOCS    _B_INTVCS       *
 83:
 84:         andi.w  #$f8ff,sr       *割り込み許可
 85:         suba.l  #$f0,a6
 86:         move.l  a6,-(sp)
 87:         DOS     _MFREE
 88:         addq.l  #4,sp
 89:         pea     mess2
 90: disp:   DOS _PRINT
 91:         addq.l  #4,sp
 92:         move.l  spsave(pc),a1
 93:         IOCS    _B_SUPER
 94:         DOS     _EXIT
 95:
 96: error:  pea     mess3
 97:         bra     disp
 98:
 99: unknown:                        *新たに常駐
100:         bsr     checksw
101:         lea.l   inter,a1        *セーバー常駐本体
102:         move.w  #$40,d1         *RTCアラーム 1Hz/16Hz
103:         IOCS    _B_INTVCS       *書き替え
104:         move.l  d0,vct          *ベクターの保存
105:         move.b  #%0000_0001,d1  *MFPに対しての割り込みの許可
106:         or.b    d1,$e88009      *申請
107:         or.b    d1,$e88015      *申請
108:         move.b  #%0000_1000,d1  *RTC 16Hz ON
109:         move.b  d1,$e8a01f      *
110:         move.l  spsave(pc),a1
111:         IOCS    _B_SUPER
112:         pea     mess
113:         DOS     _PRINT
114:         addq.l  #4,sp
115:         clr.w   -(sp)
116:         move.l  #start-idn,d0
117:         move.l  d0,-(sp)
118:         DOS     _KEEPPR
119:
120: yes_switch
121:         bsr     skip_spc
122:         cmpi.b  #'c',(a2)
123:         beq     const_set
124:         cmpi.b  #'C',(a2)
125:         beq     const_set
126:         cmpi.b  #'h',(a2)
127:         beq     help_mes
128:         cmpi.b  #'H',(a2)
129:         beq     help_mes
130:         cmpi.b  #'t',(a2)
131:         beq     timer_set
132:         cmpi.b  #'T',(a2)
133:         beq     timer_set
134:         cmpi.b  #'r',(a2)
135:         beq     pro_reset
136:         cmpi.b  #'R',(a2)
137:         beq     pro_reset
138:         pea.l   mess4
139:         bra     disp
140:
141: const_set:                              *-C Switch
142:         addq.l  #1,a2
```

```
143:          bsr     skip_spc
144:          tst.b   d6
145:          bne     no_set
146:          bsr     num
147:          cmpi.b  #2,d0               *上限 2
148:          bhi     cmax
149:          tst.b   d0                  *下限 0
150:          bmi     cmin
151: c_set:   move.b  d0,d1
152:          move.b  d0,d2               *d2=表示用
153:          add.b   d0,d0               *2ﾊﾞｲ
154:          add.b   d0,d0               *4ﾊﾞｲ
155:          add.b   d1,d0               *5ﾊﾞｲ
156:          move.b  d0,ctras
157:          addi.b  #$30,d2
158:          move.b  d2,consm2
159:          bra     checksw
160:
161: cmax:    move.b  #2,d0
162:          bra     c_set
163: cmin:    clr.b   d0
164:          bra     c_set
165: no_set:
166:          pea.l   mess6
167:          bra     disp
168:
169: timer_set:                           *-T Switch
170:          addq.l  #1,a2
171:          bsr     skip_spc
172:          tst.b   d6
173:          bne     no_set
174:          bsr     num
175:          cmpi.b  #30,d0              *上限30分
176:          bhi     tmax
177:          cmpi.b  #1,d0               *下限1分
178:          bmi     tmin
179: t_set:   move.b  d0,d3
180:          mulu.w  #960,d0             *1分当たりのカウント 60sec/(1/16
Hz)=960
181:          move.w  d0,wtime
182:          bsr     dec
183:          bra     checksw
184:
185: tmax:    move.b  #30,d0              *最大30分
186:          bra     t_set
187: tmin:    move.b  #1,d0               *最小1分
188:          bra     t_set
189:
190: pro_reset:                           *-R Switch
191:          tst.b   d6
192:          beq     no_zyochu
193:          rts
194: no_zyochu:
195:          pea.l   mess5
196:          bra     disp
197:
198: help_mes:                            *-H Switch
199:          pea     helpm
200:          bra     disp
201:
202: checksw:
203:          bsr     skip_spc
204:          cmpi.b  #'/',(a2)+
205:          beq     yes_switch
206:          cmpi.b  #'-',-1(a2)
207:          beq     yes_switch
208:          tst.b   (a2)
209:          bne     checksw
210:          rts
211:
212: num:     clr.l   d0                  *ASCII_CORD を 数値化
213:          clr.l   d1
214: num_lp:
215:          move.b  (a2)+,d1
216:          subi.b  #$30,d1
217:          bmi     num_st
218:          cmp.b   #9,d1
219:          bhi     num_st
220:          add.l   d0,d0
221:          move.l  d0,d3
222:          lsl.l   #2,d0
223:          add.l   d3,d0
224:          add.l   d1,d0
225:          bra     num_lp
226: num_st:
227:          subq.l  #1,a2
228:          rts
229:
230: dec:     divu.w  #10,d3              *16進数をアスキーコードに
231:          addi.b  #$30,d3             *アスキー変換
232:          move.b  d3,d4
233:          lsl.w   #8,d4
234:          swap.w  d3
235:          addi.b  #$30,d3             *アスキー変換
236:          move.b  d3,d4
237:          move.w  d4,timem2
238:          rts
239: skip_spc:
240:          cmpi.b  #' ',(a2)+          *空白  ポイント+
241:          beq     skip_spc
242:          cmpi.b  #',',-1(a2)         * , の場合
243:          beq     skip_spc
244:          cmpi.b  #09,-1(a2)          *又はTABの場合
245:          beq     skip_spc
246:          subq.l  #1,a2
247:          rts
248:          .data
249: ********************************** 注意 *************************
***************
250: *timem2までは、文字数に注意することアドレスエラーの原因になる(奇数バイト)*
251: ***************************************************************
***************
252:          .even
253: spsave:  dc.l    0
254: mess:    dc.b    ' >',$1b,'[35mSCREEN SAVERだ! '
255:          dc.b    $1b,'[36m             常駐開始..',$1b,'[47m^o^'
,$1b,'[33m ver1.1 By Yoneda',13,10
256:          dc.b    '       コントラストを '
257: consm2:  dc.b    '1',' に設定しました。',13,10
258:          dc.b    '       画面を暗くする待ち時間を'
259: timem2:  dc.w    '10'
260:          dc.b    '分に設定しました。',13,10,0
261: mess2:   dc.b    $1b,'[36m ..... 常駐解除  ',$1b,'[47m;_;',$1b
,'[33m',13,10,0
262: mess3:   dc.b    '既に使用されています。',13,10,0
263: mess4:   dc.b    '未定義のスイッチです。',13,10,0
264: mess5:   dc.b    'SCREENSAVERは常駐していません。',13,10,0
265: mess6:   dc.b    '解除してから再実行してください。',13,10,0
266: helpm:   dc.b    13,10,' ～  説明  ～',13,10
267:          dc.b    '  働き ... ディスプレイの焼き付きを防止する簡易型スクリーン
・セーバー',13,10
268:          dc.b    '  書式 ... Ssaver.x [/Switch]',13,10
269:          dc.b    '  Switch                      内容
設定範囲 ',13,10
270:          dc.b    '  -C num                      CONTRAST        0
 ... 真っ暗  ',13,10
271:          dc.b    '                                              1
 ... 暗い    ',13,10
272:          dc.b    '                                              2
 ... やや暗い',13,10
273:          dc.b    '  -H                          Help command',13,
10
274:          dc.b    '  -T num                      Timer set
1分から30分まで(1分刻み)',13,10
275:          dc.b    '  -R                          常駐解除',13,10,10
276:          dc.b    '       (例) Ssaver.x /t5/c0     5分後、"真っ暗"に
設定の場合。',13,10,10,0
277:          .even
278:          .end    start
```

リスト2 OBSCURE.S

```
 1: *OBSCURE.X programmed 1993,1994 by M.Kamada
 2:
 3: *ワークエリア
 4:          .offset 0
 5: rect:    .ds.w   4       *レクタングルレコード
 6: record:  .ds.b   18      *イベントレコード
 7:          .ds.b   8192    *スタックエリア
 8: workSize:                *ワークエリアのサイズ
 9:          .text
10:
11: *モジュールヘッダ
12: head:    .dc.l   'OBJR'          *モジュールタイプ
13:          .dc.l   tail-head       *モジュールサイズ
14:          .dc.l   entry-head      *実行開始位置
15:          .dc.l   workSize        *ワークエリアのサイズ
16:          .dc.l   2               *コモンエリアのサイズ
17:          .dc.l   0,0,0           *未使用
18:
19: *コマンドラインから起動
20: exec:    pea.l   notice(pc)
21:          .dc.w   $FF09   *_PRINT
22:          .dc.w   $FF00   *_EXIT
23: notice:  .dc.b   'SX-SYSTEM上で起動して下さい',13,10,0
24:          .even
25:
26: *モジュール起動
27: entry:   move.l  d0,d2   *タスクID
28:          tst.w   d1
29:          bne     dialog  *2つ目はダイアログを表示
30:          clr.w   (a4)    *解除ボタンが押されるまで0
31: event:   pea.l   record(a1)      *イベントレコード
32:          move.w  #$3001,-(sp)    *イベントマスク
33:          .dc.w   $A357   *TSEventAvail   イベント待ち
34:          addq.l  #6,sp
35:          tst.w   record+0(a1)    *イベントコード
36:          beq     idle    *アイドルイベント
37: *タスクマネージャイベント
38:          move.w  record+14(a1),d0
39:          subq.w  #1,d0   *ENDTSK
40:          bne     event   *タスクの終了ではない
41: *タスクの終了
42: exit:    clr.w   -(sp)
43:          .dc.w   $A352   *TSExit
44:
45: *アイドルイベント
46: idle:    tst.w   (a4)
47:          bne     exit    *解除ボタンが押された
48:          .dc.w   $A08F   *KBCurKbrGet
49:          move.l  d0,-(sp)        *キーボードレコード
50:          .dc.w   $A08A   *KBScan
51:          addq.l  #4,sp   *キーバッファの先頭のデータ
52:          beq     event   *キーバッファが空
53:          and.b   #$7f,d0 *キーアップコードも有効
54:          cmp.b   #$55,d0
55:          bcs     obscur  *シフト系のキーでなければ消す
56:          cmp.b   #$61,d0
```

▶モデムをあげた彼に最近電話が通じない。　寺屋　俊(22)神奈川県

```
57:          bcs      event    *XFnおよびLED付のキーは無視
58:          cmp.b    #$70,d0
59:          bcs      obscur   *シフト系のキーでなければ消す
60:          cmp.b    #$74,d0
61:          bcs      event    *SHIFT/CTRL/OPT.1/OPT.2は無視
62: obscur:  move.l   record+10(a1),-(sp)      *座標
63:          .dc.w    $A1FD    *WMFind
64:          addq.l   #4,sp    *ポインタの場所を確認
65:          subq.l   #3,d0
66:          bne      event    *コンテンツ以外は無視
67:          movea.l  a0,a2    *ウィンドウレコードのアドレス
68:          .dc.w    $A20F    *WMActive
69:          cmpa.l   a0,a2    *アクティブなウィンドウか
70:          bne      event    *違う
71:          .dc.w    $A06E    *MSObscureCsr       ポインタを消す
72:          bra      event
73:
74: *ダイアログを表示する
75: dialog:
76: *ダイアログのレクタングルを得る
77:          pea.l    rect(a1)
78:          .dc.w    $A432    *SXGetDispRect
79:          addq.l   #4,sp    *表示画面のレクタングル
80:          move.w   rect+0(a1),d0    *画面左端のX座標
81:          add.w    rect+4(a1),d0    *画面右端のX座標
82:          lsr.w    #1,d0            *画面中央のX座標
83:          sub.w    #166,d0
84:          move.w   d0,rect+0(a1)    *左端のX座標を設定
85:          add.w    #332,d0
86:          move.w   d0,rect+4(a1)    *右端のX座標を設定
87:          move.w   rect+2(a1),d0    *画面上端のY座標
88:          add.w    rect+6(a1),d0    *画面下端のY座標
89:          lsr.w    #1,d0            *画面中央のY座標
90:          sub.w    #90,d0
91:          move.w   d0,rect+2(a1)    *上端のY座標を設定
92:          add.w    #130,d0
93:          move.w   d0,rect+6(a1)    *下端のY座標を設定
94: *アイテムリストを作る
95:          move.l   #iteme-items,-(sp)       *長さ
96:          .dc.w    $A021    *MMChHdlNew
97:          addq.l   #4,sp    *再配置可能ブロックを作る
98:          move.l   d0,a3    *アイテムリストへのハンドル
99:          movea.l  d0,a0
100:         movea.l  (a0),a0
101:         lea.l    items(pc),a5
102:         move.w   #(iteme-items)/2-1,d0
103: cpItem: move.w   (a5)+,(a0)+      *ブロックへコピー
104:         dbra     d0,cpItem
105: *ダイアログを開く
106:         move.l   a3,-(sp)         *アイテムリスト
107:         move.l   d2,-(sp)         *タスクID
108:         clr.w    -(sp)    *クローズボタンなし
109:         move.l   #-1,-(sp)        *最も手前に
110:         move.w   #$0260,-(sp)     *ダイアログ用
111:         move.w   #-1,-(sp)        *可視
112:         clr.l    -(sp)    *タイトルなし
113:         pea.l    rect(a1)         *レクタングル
114:         clr.l    -(sp)    *レコードはヒープに
115:         .dc.w    $A2C3    *DMOpen ダイアログを開く
116:         lea.l    30(sp),sp
117:         movea.l  a0,a2    *レコードへのポインタ
118: *ダイアログにメッセージを書く
119:         move.l   a2,-(sp)
120:         .dc.w    $A131    *GMSetGraph
121:         addq.l   #4,sp
122:         lea.l    mess(pc),a5      *メッセージの並び
123:         moveq.l  #0,d3
124:         move.w   (a5)+,d3         *次のオフセット
125: write:  movea.l  a5,a0
126:         move.l   (a0)+,-(sp)      *ローカル座標
127:         move.l   a0,-(sp)         *メッセージ
128:         .dc.w    $A1A1    *GMShadowStrZ
129:         addq.l   #8,sp
130:         adda.l   d3,a5    *次のメッセージへ
131:         move.w   (a5)+,d3         *次のオフセット
132:         bne      write
133: *ボタンが押されるまで待つ
134:         clr.l    -(sp)
135:         .dc.w    $A2C7    *DMControl
136:         addq.l   #4,sp
137:         subq.w   #1,d0    *アイテム番号-1
138:         move.w   d0,(a4) *確認=0,解除=1
139: *ダイアログを閉じる
140:         move.l   a2,-(sp)
141:         .dc.w    $A2C6    *DMDispose
142:         addq.l   #4,sp
143:         bra      exit
144:
145: *アイテムリスト
146: items:  .dc.w    2-1
147:         .dc.w    0,0,282,102,313,122
148:         .dc.b    4,6,4,'確認',0
149:         .dc.w    0,0,240,102,271,122
150:         .dc.b    4,6,4,'解除',0
151: iteme:
152:
153: *ダイアログに書くメッセージ
154: mess:   .dc.w    4+18+2,332/2-(18/2)*6,4
155:         .dc.b    'OBSCURE.X',0,0
156:         .dc.w    4+52+2,10,30
157:         .dc.b    'キーを押したとき、マウスポインタがア'
158:         .dc.b    'クティブなウィン',0,0
159:         .dc.w    4+53+1,10,48
160:         .dc.b    'ドウの上にあったら、マウスポインタを'
161:         .dc.b    '消してしまいます。',0
162:         .dc.w    4+53+1,10,66
163:         .dc.b    'マウスを動かすか、ボタンを押すと、ポ'
164:         .dc.b    'インタが現れます。',0
165:         .dc.w    4+39+1,10,84
166:         .dc.b    'スタートアップメンテに登録して使いま'
167:         .dc.b    'す。',0
168:         .dc.w    4+32+2,25,106
169:         .dc.b    'Programmed 1993,1994 by M.Kamada',0,0
170:         .dc.w    0
171:
172: tail:
173:         .end     exec
```

リスト3 PANEL.BAS

```
10 screen 0,2,1,1:str tim,b_argv(1)
20 dim char buf(7,7),g1(6299),g2(899)
30 int x,y,zx,zy,mx,my,bl,br,pxy,endfl,etc
40 int l,tm,btm,num,d1,d2,d3,b_argc=1
50 dim str mes(2)={"NEW RECORD!",
60 "     GREAT!","    GIVE UP"}
70 dim int col(2)={248,143,56}
80 if b_argc=1 then num=5 else num=val(b_argv(1))
90 if num<2 or num>8 then num=5
100 btm=num*num*10
110 pxy=(256-num*30)/2:home(1,512-pxy,504-pxy)
120 fill(0,0,29,29,11):fill(0,0,28,28,15):fill(1,1,28,28,13)
130 get(0,0,29,29,g2):wipe()
140 /*******************
150 while 1
160 tm=0:endfl=0:etc=0
170 locate 2,0:print "TIME: 0";spc(7);"BEST TIME:";btm
180 for i=0 to num-1:for j=0 to num-1:buf(j,i)=0:next:next
190 for i=1 to num*num-1
200  repeat:x=rnd()*num:y=rnd()*num
210  if buf(x,y)=0 then buf(x,y)=i:etc=1
220  until etc=1:etc=0
230 next
240 apage(1)
250 for i=0 to (num-1):for j=0 to (num-1)
260  if buf(j,i)=0 then { zx=j:zy=i
270  } else { put(j*30,i*30,j*30+29,i*30+29,g2)
280   symbol(j*30+3,i*30+3,str$(buf(j,i)),1,1,2,127+(buf(j,i) mo
d 2)*100,0)
290  }
300 next:next
310 mouse(1):mouse(4)
320 msarea(pxy,pxy+8,pxy+num*30-1,pxy+num*30+7)
330 tim=time$
340 /*
350 repeat
360  msstat(mx,my,bl,br):mspos(mx,my):mx=mx-pxy:my=my-pxy-8
370  if bl=0 then etc=0 else if etc=0 and bl then {
380   x=mx/30:y=my/30:mx=x*30:my=y*30:etc=1
390   if x=zx and y<zy then d1=-1:d2=0:d3=-1:pnl_move_y()
400   if x=zx and y>zy then d1=1:d2=29:d3=30:pnl_move_y()
410   if y=zy and x<zx then d1=-1:d2=0:d3=-1:pnl_move_x()
420   if y=zy and x>zx then d1=1:d2=29:d3=30:pnl_move_x()
430    for i=0 to num*num-2
440        if buf((i mod num),i/num)=i+1 then endfl=endfl+1
450        if endfl<>i+1 then endfl=0:break
460    next
470  }
480  if tim<>time$ then tim=time$:tm=tm+1:locate 7,0:print tm
490 until endfl=(num*num-1) or br
500 /*
510 apage(0):fill(0,110,255,170,1)
520 if br then { etc=2
530 } else if tm<btm then btm=tm:etc=0 else etc=1
540 symbol(0,114,mes(etc),1,1,2,col(etc),0)
550 symbol(32,142,"REPLAY:L   END:R",1,1,2,col(etc),0)
560 repeat:msstat(mx,my,bl,br):until bl=0 and br=0
570 repeat:msstat(mx,my,bl,br):until bl or br
580 if br then end
590 wipe():apage(1):wipe():cls
600 endwhile
610 /**************
620 func pnl_move_x()
630  l=abs(zx-x)
640  for i=0 to l-1
650    buf(zx+d1*i,y)=buf(zx+d1*i+d1,y)
660  next
670  buf(x,y)=0
680  get(mx+d2,my,zx*30+d3,my+29,g1)
690  for i=1 to 6
700    vwait(0):put(mx+d2-d1*i*5,my,zx*30+d3-d1*i*5,my+29,g1)
710    fill(mx+d3-d1*i*5,my,mx+d3-d1*i*5+d1*4,my+29,0)
720  next
730  zx=x
740 endfunc
750 /**************
760 func pnl_move_y()
770  l=abs(zy-y)
780  for i=0 to l-1
790    buf(x,zy+d1*i)=buf(x,zy+d1*i+d1)
800  next
810  buf(x,y)=0
820  get(mx,my+d2,mx+29,zy*30+d3,g1)
830  for i=1 to 6
840    vwait(0):put(mx,my+d2-d1*i*5,mx+29,zy*30+d3-d1*i*5,g1)
850    fill(mx,my+d3-d1*i*5,mx+29,my+d3-d1*i*5+d1*4,0)
860  next
870  zy=y
880 endfunc
```

▶夕方の京浜急行線の平和島→品川方面，進行方向左側の窓を見て，思わず「ファランクス」のBGMを口ずさむのは私だけではないはずだ。　北川　亮(24)東京都

ローテク工作実験室　第5回

WaveBlaster再び

Taki Yasushi **瀧　康史**

この連載の初回を飾った「Wave Blaster接続計画」の続編です。今回は前回の不備を補いつつ，プリントパターンも起こして，より本格的に検討してみました。Wave Blasterに限らずSound Blasterのドータボードならなんでも接続できます。

はじめに

私がこれを書いている時期は，東京は急に来訪した猛暑の真っ盛り。もうヘロヘロって感じです。私の部屋には暖房器具がいっぱいありますから。冷房を入れてないと，思考が集中できないぐらいの暑さになります。暑いだけならともかく，蒸し暑いというのが温帯モンスーン気候の嫌なとこですよね。

東京電力は大丈夫だろうか？　なんて自分のウチの電気代ともども心配しつつ，できる限り編集部などにいて，冷房代を浮かせている毎日であります。

さて今月のローテクです。

なにか面白いものを期待してくれている方には悪いのですが，今月は前3回分の追補版ということにします。というのも，毎月回路を作って，それに対するランニングテスト期間が非常に短いのです。

連動電源などは，ずいぶん前から作っていたものだから，害はほとんど出尽くしている（というか，ない）のですが，できたてほやほやの回路の場合，あとからいろいろ不都合が出てきたり，もっとよい方法をふと思い立ったりするのです。いまのところ，幸い不都合というほどの不都合は出てはいませんが。

本来ならば，最低でも1，2カ月，ランニングテスト期間を設ければよいのですが，毎月するとなると，そうはいっていられないときもあるわけでして……。

というわけで，今回は，前回3回分で発生した不都合，それから私が発表後，特に気がかりだった部分をもう一度取り上げてみたいと思います。

WB接続計画

WB，ローテクを最初から読んでくれた方にはもうおわかりですね。DOS/V，PC-98地方では割と有名な，Sound Blasterのアドオンボード，Wave Blasterです。アドオンボードとはいえ，実は単なるGM音源で，なおかつMIDIを持っていたので，付加回路をつけてX680x0につけてみようという計画を実行しました。

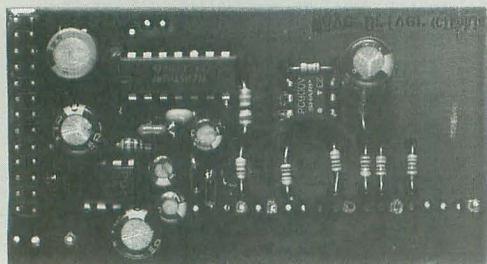
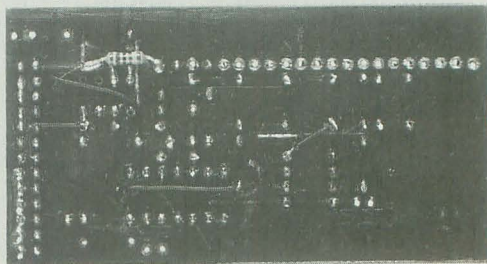
計画は見事成功しましたが，残された課題，読者からの要望がいくつかありました。

まず，RS-MIDIとしての接続回路も教えてほしいという要望。確かにWBそのものが2万円強とひたすら安いですから，RS-MIDIとしてつなげるならば，格安なMIDI音源となります。

次にプリント基板による音源基板の1枚化。電源を別に取りつけて，箱に入れ，完全にひとつの楽器として作れれば，より完成度が増します。これは要望ではなく，私自身が強く思っていたことです。

さらに，音源出力のインピーダンスマッチング。単に「鳴ればよい」という断りの下で製作を開始しましたが，なにかに負荷がかかっているのではないか？　という心配がありました。

また，どうやらWBにはMIDI-OUTが存在するという事実が判明しました。ソフトウェアデータマニュアルにはWBから送信されてくるデータがあるのに，ハードウェアシートにはないという矛盾。それをどこから得るかという問題があったわけです。有志の方のお蔭でMIDI-OUT端子がわかったため，できるならばこれにも対応したいところです。

こうなってくると，もはやもう一度見直したほうがよいのではないか？　という考えが起きました。回路の知識がちゃんとある方ならば，すでに箱にでも入れて，しっかりとした音源にしているでしょう。しかし情報としての記事だけならば，あまりにも心細いところです。良心の呵責がチクチ

プリント基板化されたWBインタフェイス

裏面，テスト基板なので一部配線は異なります

ク心をついばんできたので，再び扱うことにしました。

前回見のがした人のために，すこしWBについて説明しましょう。WBというのは，GM規格に対応した音源です。GMというのはGeneralMIDIの略，RolandのGS音源のベースとなっている世界規格です。

このGMという規格は裾野が余りにも広すぎて，イマイチはっきりしない規格です。昔というか，日本以外ではいまでもそうだと思うけど，もともとMIDI楽器は単なる「録音機材」に近いものだったので，これでことが足りたんですね。ところが最近の日本のように，「ステップエディットでごちょごちょ」する機材と考えると，規格なんてほとんど無意味ですね，私から見れば。

今度発売されたSC-88もそれなりに売れているようですが，しょせん単なるプリセットサンプラで，1年もすればプリセットされた音に飽きがくるでしょう。データコンパチビリティに重点を置かなければ，音源はシンセサイザじゃないとあまり面白くはないかもしれません。音を作るのが面倒くさい人は，サンプラか音色カードがじゃんじゃん出ているプリセットサンプラがいいですね。

私自身，音源はD70，CM-64，SC-55の順で買っていますが，飽きがきているのはSC-55だけで，CM-64は飽きるたびに新しい音色カード（いまだと4500円ぐらい）を買っているので大丈夫です。D70はシンセサイザなので，全然飽きてませんし。

WBにはカードがありません。だから，SC-55みたいに1年も使っていれば，飽きてしまうかもしれません。しかし，なによりも安さが魅力でしょう。音色カードやディスクなどでも，高いものは2万円ぐらいしますから，カード気分で新しい楽器を買っていると考えてもよいかもしれません。下手にSC-55やSC-88を買うよりも，お買い得ともいえるでしょう。

データもGMなので，探せば結構あります。X680x0のゲームで，GM対応なのはストリートファイターIIダッシュ，マッドストーカー（TG-100モードで大丈夫だと思う），あすか120%（同じくTG-100モード）などがあります。

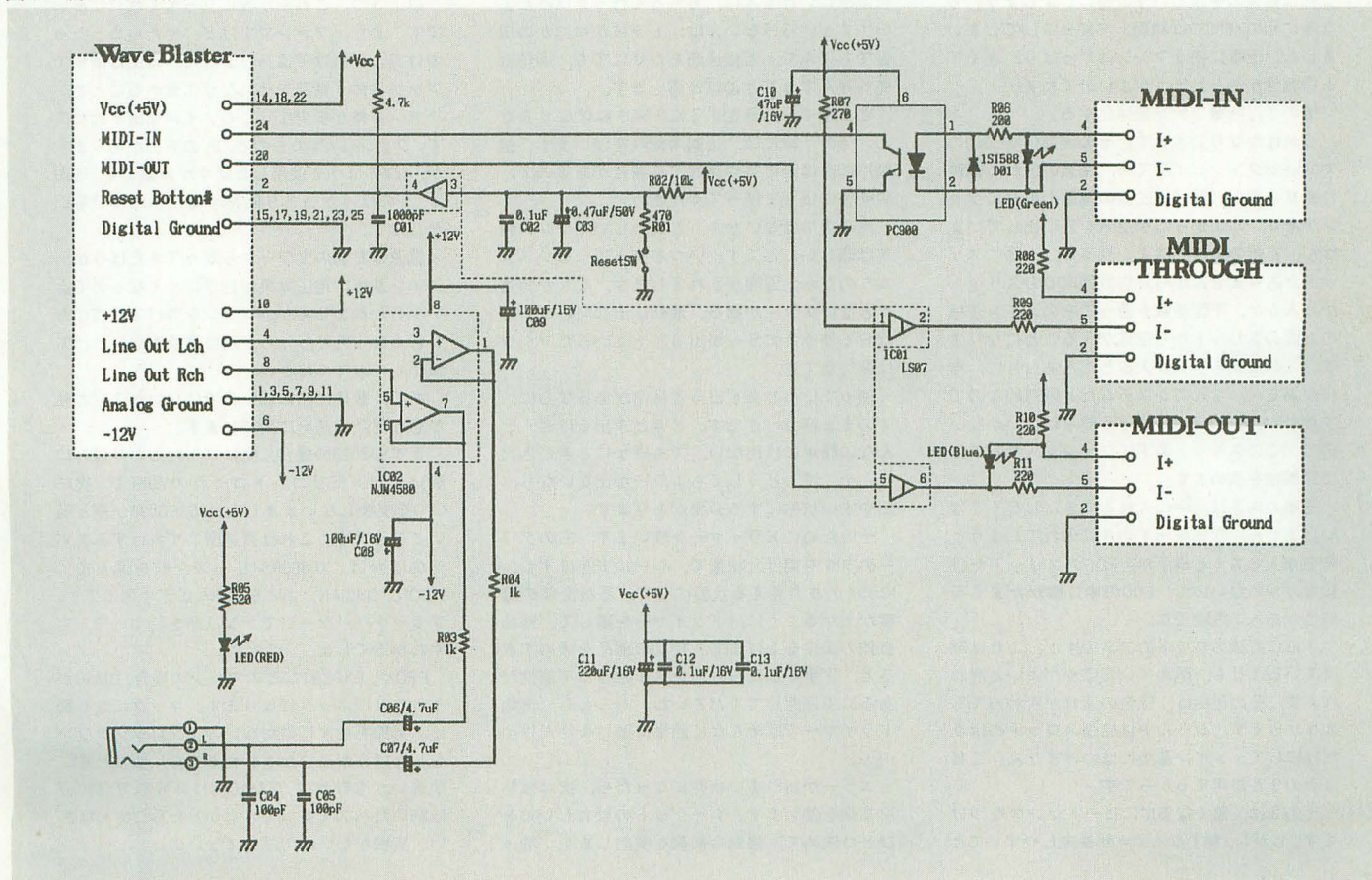
基本的にWBは日本の楽器ではなく，ROMはE-muなので，日本人ぽさがない音が詰まっています。安いという点，データ作成者にも面白い音を探すという点で魅力的でしょう。逆にいえば，ちょっとした音源がほしいという聞くだけの人にもなかなか魅力的な音源かもしれません。

ところで，噂では海の向こうにはWBとコンパチブルのインタフェイス（Sound Blasterのアドオンボード）でSC-55コンパチな音源，その名もSC-55DBが発表されたようです。詳しい情報が入手できていないのでわかりませんが，うまく輸入できれば，3万円弱で購入できるかもしれません。ということは，SC-55相当品を3万円強で手に入れられるということに……。

回路説明

図1は，今回新たに開発したWaveBlaster接続回路です。前回のWaveBlaster接続回路と比較できる方は比べてみてください。主に変わった部分はMIDI-OUTが加わった点です。

それではWaveBlaster側の端子から説

図1　新しい回路図

明しましょう。

まず，電源はVcc(+5V)と，±12Vが必要になります。VccはWaveBlasterのデジタル回路の部分に必要となり，±12Vはおそらくアナログ回路のオペアンプ部分に必要になるのだと思われます。

24pinはMIDI-INです。この信号の処理はMIDI-IN端子から，フォトカプラで電気的絶縁を図り，信号だけ伝えています。そのまま出力を，WBのMIDI信号端子に流し込み，もう片方をオープンコレクタのバッファICに流して信号を整形，および電流レベルを増強させて，MIDI-THROUGHに出力します。

MIDI-OUTはWBから出力された信号を，同じくオープンコレクタのバッファICで増強して，MIDI-OUTに出力します。LEDは図のようにつけてもつけなくても構いませんが，あると信号の入出力がわかるので便利です。

Reset Botton#の処理は，前回，単にGNDに出力を落としてローレベルにしていただけですが，今回はちゃんとした回路でサポートしています。この回路は簡単にいえば，遅延回路です。

リセット端子は，確かにGNDレベルに落としただけで問題ないはずなのですが，製品によっては，ある一定の時間，ローレベルに落とさなくてはいけないという条件があるので適当にコンデンサを使って処理しています。本当はバッファICがオープンコレクタでないほうがよいのですが，たったひとつの回路のためだけに，オープンコレクタじゃないバッファICをつけるのは経済的じゃなかったので，コレクタをVccに上げて処理しています。ひとつのバッファ

高速マシンで夏を乗り切る方法

もう9月号ですが，まだ残暑が厳しい季節でしょう。その暑さにあなたのマシンは負けていませんか？

うちの子に限って……のように，我がマシン，あなたのマシンも割と夏の熱い日に負けていることがあります。うちのマシンも冷房がついていないと勝手にハングってしまって困ってしまったことが何度かあります。

そこで，私が普段利用している，高速マシンで夏を乗り切る方法を伝授しましょう。

まずひとつ。

クロックダウンをする……ってこれでは意味ないですね。いちばんよい方法なんですけど，これは最後の手段ということにしましょう。ちなみに私のX68030は結局，25MHzにしてしまいました。仕事に使うマシンはやっぱり，遅くても信頼性があったほうがよいのでねぇ……。

その2。内蔵ファンを強化する。

これはかなり効きます。その理由はX680x0のマンハッタンシェイプでの，空気の流れる構造にあります。X680x0は左側の電源の部分にファンがあり，ここからしか空気を外に出していません。左側のタワーはよく見ると，上面にスリットがありませんから，空気はFDDのスリットから入るか，下面基板を通って右のメイン基板の上面のスリットから空気が入ることになります。この上から空気が入るところあたりに，冷房がある……なんてシステムだと理想的なのですが(うちのシステム)，そう簡単にはいかないので，このスリットの下にファンを入れることで吸風性を高めます。

送風の向きは，中に入れる向きにしなくてはいけません。空気を出す向きに入れてしまうと，空気が入ってくる場所が，FDDのスリットだけになりかねないので，FDDの中に塵がたまる可能性があって危険です。

さらに拡張スロットのフタですが，これは開けているよりも，閉めていたほうがよいと思われます。その理由は，空気の流れが変わってしまうからです。シールドは拡張スロットのほうだけにして，メイン基板にはつけません。これはその3と併用するからです。

その3は，熱くなるICにヒートシンクをつけます。しかし，熱くなるIC＝熱暴走しやすいICというわけではありません。たとえば，X68030だとYUKIがあまり熱くないうちから異常動作し始める傾向があるようですし，ドットクロックオシレータのデータセレクタ＆分周器であるOCIANは，かなり熱を持つ割には，そんな簡単には熱暴走しません。各ゲートアレイが，なにを行っているかで，熱暴走しやすいかはなんとなくはわかりますが，決定的なことはわかりません。ただ，熱を持つICが暴走しなかったにしても，まわりのICを間接的に暖めていることは事実なので，小さなヒートシンクなどを購入して，取りつけておくことは必要でしょう。薄めのヒートシンクが最近秋葉原などで出回っています。秋葉原にいける人は，もちろん行ったほうがよいですが，行けない方は，トラ技かなにかの広告でも読んで，若松通商あたりにでも，通信販売を頼んでみるとよいと思います。

ともかく，熱暴走するICを探さねばなりません。そのためには，強制冷却剤を使います。強制冷却剤は静電気や結露する場合があるので，説明書に従って使ってください。

熱暴走ではないか？　と疑わしいマシンの特定は難しいことです。いつも使っているシステムなのだが，画面が乱れてしまう，なぜか特定できないエラーが出る。最初は出ないけど，しばらく使うとエラーが出る。そういったマシンは要注意です。

使っていると必ず出ると自信があるならば，そのまま待つべきです。ときどき出るけど，そんなに簡単には出ない。でも確かにときどき出る。そこで，どうしてもエラーが出ないなら，出やすい状態にする必要があります。

そのためにドライヤーを使います。右のタワーのフタを開けた状態で，シールドをはずし，ICがそのまま見える状態にして，基板全体の温度が上がるように，ドライヤーを離して，部品自体の温度を上げます。特定の部品を暖めすぎると，下手をしたら，ハンダが融ける可能性があるので注意してくださいね。もっとも，市販ドライヤーではそんなに簡単に融けませんけど……。

エラーが出やすい状態になったら，次は強制冷却剤を使います。ターゲットの疑わしいICをひとつ決めて，強制冷却剤を噴射します。触って冷たくなるぐらいまで冷やしたのに，まだエラーが出るならば，そのICはシロです。ホシを探すまでこれを繰り返し，ターゲットを絞ります。

ターゲットが定まり，熱暴走しやすいものがわかったら，それにヒートシンクを張りつけます。単にヒートシンクを張りつけても，ほとんど効果がないので，できるならば小さなファンをヒートシンクに直接風が当たるような場所に取りつけます。風が当たるか当たらないかで，全然効率が違うので，ぜひ，2番の方法と兼ね合わせて，空気の流れる道，それにちょうど当たるヒートシンクといった具合にセットしてください。

以上3つ。これらが役に立ってくれれば幸いです。また，ファンですけど，やたらめったらつけてよいわけではありません。理由は簡単で，ファン自体が電源をそれなりに食べることと，ファン自体が多少なりともノイズを出すためです。ファンは12Vのものと，5Vのものがありますが，12Vのほうを使用したほうが無難です。理由は12Vのほうがあまり回路中で使われていないからです。

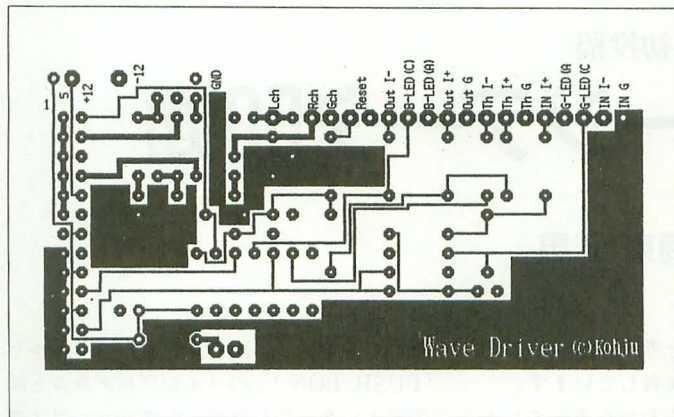
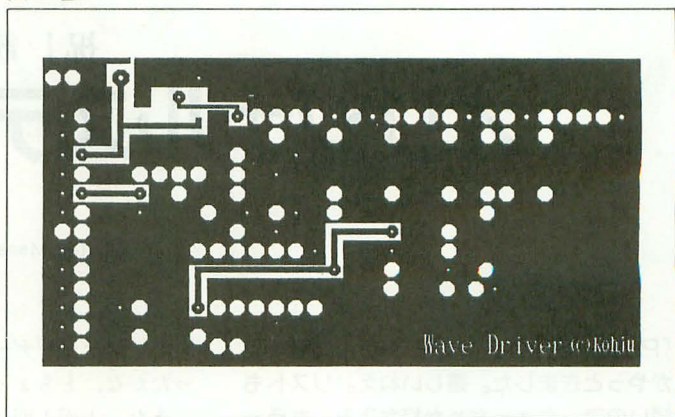
電源は左側のタワーから取ってきたほうが，メイン基板に電圧効果が起きにくくなるのでよいかもしれませんが，ファンをつけて動作がおかしくなったりしたら，別に電源を取りつけて強化してみてください。

さて，最後に熱に弱いのではないか？　と私が思う部品を適当に列挙します。

まずX68030の場合，先にいったとおり意外に弱いのがメモリコントローラのYUKIで，次にCPUを交換しないままだとMC68EC030が割と弱いところです。これは高速型にすればすみます。そのほかは，X68000シリーズ全般を通して，VICON，OSCIAN，あとSONYのビデオ用ICです。フラットパッケージでかなり熱を持つので，すぐわかるでしょう。

PROや，EXPERTなどのマシンの場合，DMACがそれなりにネックになります。ネックになる割に，過熱もせずに動かないのでクロックダウンをしたほうがよいかもしれません。私の記憶に間違いがなければ，DMACだけ非同期でX680x0は動いたハズです（できなかったらごめんなさい。実機がないので試してないの)。

図2-A　プリントパターン(原寸)

図2-B

ICの中にはたいてい6回路ぐらい入っているので，使い切ったほうがもったいなくないですしね。

4,8の出力は前回，右，左が逆という話があったので，今回は逆にしています。AUX出力をNJM4580を使い，ボルテージフォロアで信号増強を行っています。直列に1kΩの抵抗が入っていますが，これを適当に変えると出力のレベルが変わります。余裕がある人は，これを可変抵抗にしてメインボリュームにするのがよいと思います。

C06,C07は直流負荷を防ぐためのアイテムです。100pFはローパスフィルタですね。これは回路の安定のために入れています。

部品説明

図2は，プリントパターンです。部品配置は写真を見てください。それでは部品を説明します。

1．は，WB本体です。これは当然なくてはいけません。Sound Blasterとのインタフェイスがあるならば，なんでもよいですね。

2．は電源です。

3．はWBと接続できる13行×2列のコネクタです。

4．の抵抗はそれぞれ必要な分だけ集めてもよいのですが，100個単位で買っても悪くないかもしれません。1個10円，100個100円というのは，ざらにある話ですから。W数ですが，おそらく1/8でも大丈夫だと思われます。1/4あれば完璧ですね。プリント基板のサイズは，1/4サイズでも入るように作られています。

5．のコンデンサも，必要な分だけ集めてください，多少多めに買っても安いものだから構わないかもしれませんが。

6．のダイオードはMIDI端子から流れてくる負のレベルの信号を除去するためのものです。これもなくてはいけません。

7．のフォトカプラも必ず必要です。MIDI規格で推奨されているフォトカプラらしいですが定かではありません。

8．のLEDは，赤，緑，青と書いています。ですが，別に3つあればなんでも構いません。3つ同じ色だと，どれがどれだか，わからなくなってハマるので，多分，違う色のほうがよいでしょう。青のMIDI-OUTはおそらく滅多に光らないんじゃないか？という条件からつけています。麗しの青は貴重なほうがよいのです。

9．の5ピンDINはMIDI出力端子用です。今回は出力まであるので3つ必要です。

10．のAUX出力端子は，MINI PHONEでも，PHONE端子でもAUX-PIN JACKでも好きなものを使ってください。

11．オペアンプはNJM4580です。別にこれを選んだ理由はありません。ただうちで余っていただけです。

12．のリセットボタンはケースにフィットしたものを買いましょう。単なるスイッチで構いません。

13．の電源ON/OFF用スイッチは主電源のためのスイッチです。スイッチング電源はAC側で操作するので，141V/5Aぐらいの耐圧があったほうがよいでしょう。これは前回つけずに編集部でハマったものです。

14．はケース。うまく加工して，綺麗に仕上げてくださいね。

まとめ

これでWBインタフェイスの作成は終了です。結果的にCM-64相当のインタフェイスまで持てたので，これは（バグがない限り）これでおしまいにしたいと思います。

RS-232C-MIDIについてですが，RS-232C端子で接続はできたものの，イマイチ納得のいくものができなかったので，今回はおあずけです。

具体的に私が満足いくデキで，できたならばこの連載のコラムとしてでも発表したいと思います。

Special Thanks：P.E.I.氏（in XICLUB）

部品表

1．Wave Blaster本体
2．電源＋5V，±12V
3．ジャンパスイッチを接続できる13×2列のコネクタ
4．抵抗
　270Ω　1つ
　220Ω　4つ
　200Ω　4つ
　470Ω　1つ
　10kΩ　4つ
　4.7pΩ　1つ
　1pΩ　2つ
　520Ω　1つ
5．コンデンサ
　47μF/16V(電解)　1つ
　パスコン用0.01μF　1つ
　220μF/16V(電解)　1つ
　100PF　2つ
　4.7μF/16V(電解)　2つ
　100μF/16V(電解)　2つ
　1000PF　1つ
　0.1μF　1つ
　0.47μF/50V(電解)　1つ
6．ダイオード
　1S1588　1つ
7．フォトカプラ
　PC900
8．LED　赤、緑、青　それぞれ1つ
9．5ピンDIN（MIDI端子用）　3つ
10．AUX出力用コネクタ　1つ
11．オペアンプ（NJM4580）　1つ
12．リセットボタン　1つ
13．電源ON/OFF用スイッチ　1つ
14．ケース　1つ

PUSH BON!

祝！ 読者初投稿

オリジナルステージデータ50面

Shutou Masao 周東 正男

「PUSH BON!」のオリジナルステージがやっときました。嬉しいねえ。リストも短いので，ちゃっちゃか打ち込み，もう一度，50面分ハマってみませんか？

う〜ん50面

こんにちは。「PUSH BON!」オリジナルステージ募集ということで，気合一発投稿しました。最初は，20面くらいだったのに，いろいろと考えているうちに40面となり……そうなると全面作りたくなるのが人情というもの。結局，50面全部作ってしまいました。

さて，今回オリジナルステージを考えるときには，まず解き方を初めに考えたので最小ステップ数は少なめのステージが多くなってしまいました。もう少し，違う見方をしてステージデータを考えたほうがよかったかな，とちょっと反省しています。

また，1面1面のタイトルも考えてしまいました。タイトルの由来は，見た目もありますが，解き方のヒントを表している場合もあります。参考にしてください。そして，タイトルの横には私のベストステップ数も記しておきます。もしも最小ステップを更新するようなことがあれば，教えてほしいな。

リストは，MAC.Xのファイル入力ツールを使って打ち込み，セーブバイト数783バイトでセーブを行ってください。ファイルを作成できたら，

LHA E PUSH.LZH

として，

PUSH_USR!.MAP

というユーザーマップデータファイルを解凍してください。次に，できたファイルを，「PUSH BON！」のメインプログラムと同じディレクトリにコピーして，コンフィグでユーザー定義のマップで遊べるように設定すれば，このオリジナルマップを遊ぶことができます。

もしくは，オリジナルステージデータファイル（PUSH_BON！.MAP）そのものをいったんリネームし，解凍されたPUSH BON！.MAPにリネームすればOKです。この場合は，コンフィグで設定を変えることなく遊べます。

なお，以前に自分でエディットしたステージデータがある場合は，あらかじめマップデータファイルをリネームしておいてください。

全面解けることは確認してありますので，ぜひとも挑戦してみてください。

リスト

```
0000  26 AF 2D 6C 68 35 2D E6  : 1E
0008  02 00 00 92 09 00 00 91  : 2E
0010  9A E7 1C 20 01 0D 50 55  : 70
0018  53 48 5F 55 53 52 21 2E  : 43
0020  4D 41 50 03 5C 48 00 00  : 85
0028  02 36 6B 12 69 C8 23 89  : 92
0030  C0 03 1C C3 1B 26 CE B6  : 67
0038  AB E9 7E EF 74 B2 36 D3  : 30
0040  F7 33 E9 E3 DE 31 9F AF  : 53
0048  9A 35 98 BA D3 1E 32 B9  : FD
0050  3C 78 78 59 72 22 F9 2D  : 3F
0058  C7 7B E7 B0 F5 1D 87 76  : E8
0060  6B F5 95 CF 1E 75 3D CA  : 5E
0068  1A 07 32 FE 1E AC EC 3A  : 41
0070  37 B8 16 23 86 55 EC F9  : E8
0078  E4 99 E1 E5 33 47 53 3B  : 4B
------------------------------------
SUM:  03 E9 9B B5 26 C7 7E 4F  2A8D

0080  C9 A1 7F 37 E5 A0 E5 5F  : E9
0088  D6 CC 0D 2A E7 CE 5B F3  : DC
0090  DC F5 A8 D4 F3 3A 81 68  : 63
0098  5A 85 42 BE 83 D3 C8 11  : 0E
00A0  05 52 18 4A C6 76 52 1E  : 65
00A8  A6 03 84 39 CF AB 6C 72  : BE
00B0  A0 A0 E2 12 EE 4C 0F A0  : 1D
00B8  E8 63 14 F8 8D 6C DB BD  : E8
00C0  A9 32 19 D7 5D 36 7F DF  : BC
00C8  55 07 B5 42 65 26 B9 A1  : 38
00D0  A0 3B 52 7A 29 8D A5 32  : 34
00D8  FD 00 A4 7F 6F A2 47 17  : 8F
00E0  19 20 4C 3D A3 24 F6 22  : A1
00E8  47 60 F9 80 F3 CB 89 5C  : C3
00F0  D9 07 8A D9 A7 B9 D3 3A  : B0
00F8  11 1C AF 32 17 74 25 BF  : 7D
------------------------------------
SUM:  ED 56 4A 5A 00 FB CC F8  D4EF

0100  D1 87 F6 52 63 55 49 C6  : 67
0108  F3 CC 4A 31 94 AE D7 0E  : 61
0110  4C 79 E3 F5 D7 52 9F E9  : 4E
0118  89 83 56 17 6D 77 C7 8C  : B0
0120  36 76 E9 4C 4B 52 F3 E0  : 51
0128  4B 1D 4A CF 48 29 81 95  : 08
0130  9F 97 1A E6 59 BC 14 9E  : FD
0138  E6 B2 00 C3 FA 74 60 58  : 81
0140  DA 02 53 43 40 71 A7 C1  : 8B
0148  7F 98 1E B5 D5 D0 92 50  : 71
0150  AF FE 0B 14 83 90 39 2C  : 44
0158  87 57 B6 A2 BC 4B F5 6E  : A0
0160  BF 71 DF 62 36 4C BB 95  : 43
0168  92 33 E0 11 85 CA 1E E3  : 06
0170  89 70 29 A7 A1 C2 FD 76  : 9F
0178  27 43 E3 E4 4F 04 1C 98  : 38
------------------------------------
SUM:  2F 71 C3 FF 20 6F C7 E5  C2C7

0180  1D 42 74 C2 99 F2 2B 0D  : 58
0188  AC 41 37 CC CD 90 9B A1  : 89
0190  AC 3E 60 3E 1E 03 C8 69  : DA
0198  67 E4 4A 65 99 0D FB D4  : 6F
01A0  C0 F4 F8 3F EA 30 66 8E  : F9
01A8  D0 86 D2 32 FA 5C 09 31  : EA
01B0  1F 09 F6 24 5A 0C 7D 14  : 39
01B8  60 92 4A D1 88 A1 4B 6F  : F0
01C0  F4 A0 55 5A DE AA 77 53  : 95
01C8  32 B9 D2 F1 46 6A D4 F7  : 29
01D0  2B F8 6A 5B 30 3D 5F 2A  : DE
01D8  07 6C 7E 42 8F D2 DC BB  : 2B
01E0  8E 5F 52 F8 1D 1F B4 E5  : 0C
01E8  48 A8 51 FA B4 34 7F F9  : 9B
01F0  1A 73 C1 A1 7E 98 59 0D  : 6B
01F8  48 11 75 86 46 EA FC E6  : 66
------------------------------------
SUM:  7B 02 47 98 5B C3 CE 2D  C64F

0200  A4 38 74 19 A7 F0 13 52  : 65
0208  3B B4 9F CE 44 6A 82 11  : 9D
0210  6A 8F 04 73 01 D9 B3 3B  : 38
0218  9C 4B DB 6C 48 5B 81 21  : 73
0220  7F 3E 8E 5E 81 2F B1 E9  : F3
0228  33 F7 7D 63 D3 A7 C2 BA  : 00
0230  74 CF F8 B3 E1 80 C9 FB  : 13
0238  3C F1 D2 36 14 6A DA 5A  : E7
0240  BE E8 87 D3 2C 98 F3 E9  : A0
0248  05 A7 54 3C 28 2D 4E D3  : B2
0250  5F 68 62 1F C1 DD 7F F8  : 5D
0258  A3 1B 08 93 38 D8 50 64  : 1D
0260  F7 3F 31 F5 B2 9B A9 BA  : 0C
0268  ED 87 E7 F9 C5 95 11 51  : 10
0270  BF 31 8A 3A 7D D7 5A 3D  : 9F
0278  67 65 48 EC 75 32 E6 9C  : 29
------------------------------------
SUM:  16 29 F6 45 33 01 E9 B3  B95D

0280  F4 70 21 BD 8E FD 68 7E  : B3
0288  AF 58 88 DA C0 06 B1 20  : 00
0290  F6 1A 2A 3C F8 10 38 BD  : 73
0298  EE 19 2B C6 C2 58 A4 23  : D9
02A0  9A 60 53 A8 89 28 A0 28  : 6E
02A8  B6 8E 0B 93 BD 88 31 9C  : F4
02B0  11 EB 93 EE 7B D0 14 3F  : 1B
02B8  2D 3B 9A 7C 37 D4 FB 64  : E8
02C0  FE 76 8B E6 11 3E DD AF  : C0
02C8  94 B1 02 7A 27 B7 DE F0  : 6D
02D0  AC 9F AB 50 C6 23 5B DE  : 68
02D8  D8 36 62 9B F3 D1 F4 F6  : B9
02E0  A0 52 E9 4C F5 44 3E D8  : 76
02E8  18 8B AE 76 B4 3D 9C BD  : 11
02F0  3C 5A E1 C6 B4 19 85 87  : 16
02F8  D8 AE FF 87 AB 8E BC B5  : B6
------------------------------------
SUM:  F7 F0 9A 98 F9 D0 FA 29  3126

0300  F2 FF 24 CC AC 34 15 60  : 36
0308  5B ED E6 32 E9 A0 00 00  : E9
0310  00 00 00 00 00 00 00 00  : 00
0318  00 00 00 00 00 00 00 00  : 00
0320  00 00 00 00 00 00 00 00  : 00
0328  00 00 00 00 00 00 00 00  : 00
0330  00 00 00 00 00 00 00 00  : 00
0338  00 00 00 00 00 00 00 00  : 00
0340  00 00 00 00 00 00 00 00  : 00
0348  00 00 00 00 00 00 00 00  : 00
0350  00 00 00 00 00 00 00 00  : 00
0358  00 00 00 00 00 00 00 00  : 00
0360  00 00 00 00 00 00 00 00  : 00
0368  00 00 00 00 00 00 00 00  : 00
0370  00 00 00 00 00 00 00 00  : 00
0378  00 00 00 00 00 00 00 00  : 00
------------------------------------
SUM:  4D EC 0A FE 95 D4 15 60  77B6
```

一挙公開!
オリジナルステージデータ全50面

ステージ番号：ステージタイトル(最小ステップ数)

1：X(5)　2：Wave(2)　3：エレベーター(20)　4：ノーマル面(7)

5：マイマイ(7)　6：Crash1(6)　7：ノーマル面2(7)　8：Crash2(3)　9：Wave2(3)　10：Crash3(4)　11：ノーマル面3(11)

12：マイマイ2(4)　13：リトルスター(8)　14：エルドラド(10)　15：ノーマル面4(14)　16：栓抜き(12)　17：Bingo!!(4)　18：House!!(13)

19：マイマイ3(6)　20：ペンゴ風(8)　21：Yマーク(11)　22：コア1994(10)　23：エレベーター2号(26)　24：あみだくじ(2)　25：トリック(10)

26：ノーマル面5(10)　27：大円舞(8)　28：エレベーター3号(26)　29：あみだくじ2号(3)　30：ストライク!!　(5)　31：007(10)　32：鍾乳洞(7)

33：名月(2)　34：十字架(2)　35：ノーマル面6(17)　36：36(12)　37：ノーマル面7(6)　38：P/R(7)　39：ペンゴ風2(8)

40：FM(9)　41：AM(11)　42：Down Load(11)　43：モーゼの十戒(15)　44：十字架2(4)　45：YMO(19)　46：愛の十字架(13)

47：ノーマル面8(16)　48：迷走(9)　49：ノーマル面9(11)　50：The End(25)

さすがに50面一挙掲載すると壮観だなあ。とぼんやり眺めているそばから，さらに2人の読者からオリジナルステージデータの投稿がありました。こちらについては，10月号に予定している付録ディスクに収録するつもりです。結局，総計140面分というステージデータが投稿されました。また眠れない日々が続いちゃうかな。

PUSH BON!

「PUSH BON!」なんてラクチンさ

ステージデータ自動解法プログラム

Kamata Makoto 鎌田 誠

あっと驚く「PUSH BON!」ステージデータ自動解法プログラム。ステージの確認用として使ってください。解けないからといってプログラムに頼ってはいけませんよ。

オートでラクラク

「こいのぼりPRO-68K」に収録されていたパズルゲーム「PUSH BON!」の各ステージを，最小何ステップで解くことができるか，厳密に調べてみました。

残念ながら，現在ステージ36について最小のステップ数が求まっていません（編注：後日，すべての解法が送られてきました）。ステージ35までは手で解けたのですが，ステージ36で詰まってしまったので，ステージ36以降も含めた最小ステップ数の探索に移りました。ステージ36が解けていないのにステージ37以降の最小ステップ数が求まっているのは，マップファイルを見たからです。

最小ステップ数の探索には主にX68000 PRO（無改造）を用い，特に最小ステップ数の多いステージについては研究室のワークステーションを使用しました。

右側にある表にこのプログラムで求めた最小ステップ数を示します。なお，ステージ36については現時点までの探索で得られた最小ステップ数の範囲を書いてあります。

下にある図に，プログラムが出力する最小ステップ数での解答例を列挙します（ステージ1）。初期状態からスターブロックが3個並ぶまで，プレイヤーまたはブロックのいずれか1つが一方向に移動するたびに，ステージ全体を描き直しています。ブロックを押したときのプレイヤーの向きは示されていないので注意してください。ここに収録した解答例は，コンピュータによって解かれた手順を視覚化したものです。LRブロックを使った連鎖的な移動の回数を最適化していないので，冗長な連鎖を含んでいる場合があります。記号は以下のように対応しています。

記号	意味
◎	プレイヤー
・	空き
□	ノーマルブロック
■	固定ブロック／壁
L	Lマークブロック
R	Rマークブロック
☆	スターブロック

では，プログラムの使い方です。プログラムは2つに分かれています。リスト1は，マップファイルからステージデータを切り出すためのものです。とりあえず，X-BASICで打ち込み，“RUN”で実行してください。実行して，「PUSH BON!」のマップファイル名を入力し，切り出したいステージ番号を入力すると，「STAGE ??.MAP」というテキストファイルが出力されます。

次に，リスト2をコンパイルし（GCCでのみ動作確認済み），コマンドラインから，

SOLVE STAGE??.MAP

のようにして実行してください。すると標準出力に解析結果を出力していきます。ファイルに残したい場合は，

SOLVE STAGE??. MAP > STAGE??.OUT

のようにして，リダイレクトを使うようにしましょう。なお，オプションとして，

SOLVE マップファイル名 ［最小手数 最大手数 反射回数］

以上のような設定もできますが，省略されると，最小手数=1，最大手数=100，反射回数=10のようにして実行されます。

解析には，手数が多くなるにつれて実行時間がかかるようになります。気長に待ってやってください。

図 出力例

```
マップファイル: stage1.map
最小手数: 1
最大手数: 100
反射回数: 10

問題:
■■■■■■■■■
■◎・・・・・・■
■・・・☆・・・■
■・□・・・□・■
■・・・☆・・・■
■・□・・・□・■
■・・・☆・・・■
■□・・・・・□■
■■■■■■■■■

1 手で挑戦します
2 手で挑戦します

2 手で解けました

バックトレースします
■■■■■■■■■
■・・・◎・・・■
■・・・・・・・■
■・□・☆・□・■
■・・・☆・・・■
■・□・☆・□・■
■・・・・・・・■
■□・・・・・□■
■■■■■■■■■

■■■■■■■■■
■・・・◎・・・■
■・・・☆・・・■
■・□・・・□・■
■・・・☆・・・■
■・□・☆・□・■
■・・・・・・・■
■□・・・・・□■
■■■■■■■■■

■■■■■■■■■
■・・・・・・・■
■・・・☆・・・■
■・□・・・□・■
■・・・☆・・・■
■・□・☆・□・■
■・・・・・・・■
■□・・◎・・□■
■■■■■■■■■

■■■■■■■■■
■・・・・・・・■
■・・・☆・・・■
■・□・・・□・■
■・・・☆・・・■
■・□・・・□・■
■・・・☆・・・■
■□・・◎・・□■
■■■■■■■■■

■■■■■■■■■
■◎・・・・・・■
■・・・☆・・・■
■・□・・・□・■
■・・・☆・・・■
■・□・・・□・■
■・・・☆・・・■
■□・・・・・□■
■■■■■■■■■
```

表　プログラムを使って求めた最小ステップ数（ステージ，最小ステップ数）

ステージ	最小ステップ数	ステージ	最小ステップ数	ステージ	最小ステップ数	ステージ	最小ステップ数	ステージ	最小ステップ数	ステージ	最小ステップ数
1	2	10	4	19	2	28	18	37	6	46	1
2	4	11	6	20	2	29	7	38	5	47	7
3	5	12	5	21	6	30	9	39	5	48	9
4	6	13	4	22	3	31	13	40	6	49	7
5	2	14	10	23	5	32	6	41	6	50	16
6	2	15	4	24	4	33	6	42	5		
7	4	16	5	25	4	34	8	43	6		
8	4	17	8	26	11	35	4	44	6		
9	5	18	5	27	7	36	29?	45	5		

リスト1　MAP CUT.BAS

```
10 /*
20 /*マップデータの切り出しプログラム
30 /*
40 int fp1,fp2,i,j,num
50 int stage=0
60 str fn,w
70 str waku="■■■■■■■■■"
80 str waku2="■"
90 str my_chr="◎"
100 dim str map_chr(5)={ "・","□","L","R","☆","■" }
110 /*
120 input "マップファイルを入力してください",fn
130 while stage<1 or stage>50
140   input "ステージ番号を入力してください(1-50)",stage
150 endwhile
160   fp1=fopen(fn,"r")
170   fp2=fopen("STAGE"+right$(itoa(stage),2)+".MAP","c")
180   stage=stage-1
190   fseek(fp1,stage*49,0)
200   fwrites(waku,fp2)
210   fputc(&HD,fp2)
220   fputc(&HA,fp2)
230   for i=0 to 6
240    fwrites(waku2,fp2)
250    w=""
260    for j=0 to 6
270      num=fgetc(fp1)
280      w=w+map_chr(num)
290    next
300    fwrites(w,fp2)
310    fwrites(waku2,fp2)
320    fputc(&HD,fp2)
330    fputc(&HA,fp2)
340   next
350   fwrites(waku,fp2)
360   fputc(&HD,fp2)
370   fputc(&HA,fp2)
380   fwrites("ステージ番号   "+itoa(stage+1),fp2)
390   fputc(&HD,fp2)
400   fputc(&HA,fp2)
410   fseek(fp2,22,0)
420   fwrites(my_chr,fp2)
430 fcloseall()
440 end
```

リスト2　SOLVE.C

```
  1: /*
  2:   パズルゲーム"PUSH-BON!"を解くプログラム
  3:   1994.04.22-1994.05.08 by M.Kamada
  4: */
  5:
  6: #include <stdio.h>
  7:
  8: #define SIZE_X 7     /* ステージの大きさ */
  9: #define SIZE_Y 7
 10:
 11: #define MAX_BOUND 10     /* 反射回数の最大値 */
 12:
 13: #define MIN_TRY 1     /* ステップ数の最小値 */
 14: #define MAX_TRY 100     /* ステップ数の最大値 */
 15:
 16: #ifdef __human68k__
 17: /* X680x0用の定義 */
 18:
 19:   typedef unsigned short stone;     /* ブロックを2バイトで表現 */
 20:
 21: /* ブロックを表すキャラクタ */
 22: #  define STAR L'☆'
 23: #  define PLAYER L'◎'
 24: #  define SPACE L'・'
 25: #  define NORMAL L'□'
 26: #  define LEFT L'L'
 27: #  define RIGHT L'R'
 28: #  define WALL L'■'
 29:
 30: /* ブロックをファイルから読み込むためのマクロ */
 31: #  define FSCANF_STONE(fp,x) (fscanf(fp, "%2c", &(x)))
 32: /* ブロックを表示するためのマクロ */
 33: #  define PRINTF_STONE(x) (printf("%c%c", (x) / 256, (x) % 256))
 34:
 35: /* メッセージ */
 36: #  define M_ILLEGAL "の指定が違います¥n"
 37: #  define M_USAGE "パズルゲーム¥"PUSH-BON!¥"を解くプログラム¥n¥
 38: 使い方: %s マップファイル [最小手数 [最大手数 [反射回数]]]¥n"
 39: #  define M_MAP_FILE "マップファイル"
 40: #  define M_MIN_TRY "最小手数"
 41: #  define M_MAX_TRY "最大手数"
 42: #  define M_MAX_BOUND "反射回数"
 43: #  define M_QUESTION "問題:¥n"
 44: #  define M_TRY_IN "%d 手で挑戦します¥n"
 45: #  define M_CANT_IN "%d 手以内では解けません¥n"
 46: #  define M_SOLVED_IN "¥n%d 手で解けました¥n"
 47: #  define M_BACK_TRACE "¥nバックトレースします¥n"
 48: #  define M_FILE_NOT_FOUND "ファイルが見付かりません"
 49: #  define M_FORMAT_ERROR "フォーマットエラー"
 50: #  define M_ILLEGAL_CHARACTER "無効なキャラクタがあります"
 51: #  define M_MISSING_PLAYER "プレイヤーが指定されていません"
 52: #  define M_TOO_MANY_PLAYERS "プレイヤーが複数指定されています"
 53: #  define M_MISSING_STAR_BLOCK "スターブロックが足りません"
 54: #  define M_TOO_MANY_STAR_BLOCKS "スターブロックが多すぎます"
 55:
 56: #else
 57: /* X680x0以外のマシン用の定義 */
 58:
 59:   typedef unsigned char stone;     /* ブロックを1バイトで表現 */
 60:
 61: /* ブロックを表すキャラクタ */
 62: #  define STAR 'S'
 63: #  define PLAYER 'P'
 64: #  define SPACE '.'
 65: #  define NORMAL 'N'
 66: #  define LEFT 'L'
 67: #  define RIGHT 'R'
 68: #  define WALL 'W'
 69:
 70: /* ブロックをファイルから読み込むためのマクロ */
 71: #  define FSCANF_STONE(fp,x) (fscanf(fp, "%1c", &(x)))
 72: /* ブロックを表示するためのマクロ */
 73: #  define PRINTF_STONE(x) (printf("%c", (x)))
 74:
 75: /* メッセージ */
 76: #  define M_ILLEGAL " error¥n"
 77: #  define M_USAGE "Puzzle game ¥"PUSH-BON!¥" solver¥n¥
 78: usage: %s map-file [min-try [max-try [max-bound]]]¥n"
 79: #  define M_MAP_FILE "map-file"
 80: #  define M_MIN_TRY "min-try"
 81: #  define M_MAX_TRY "max-try"
 82: #  define M_MAX_BOUND "max-bound"
 83: #  define M_QUESTION "question:¥n"
 84: #  define M_TRY_IN "try in %d¥n"
 85: #  define M_CANT_IN "can't solve in %d¥n"
 86: #  define M_SOLVED_IN "¥nsolved in %d¥n"
 87: #  define M_BACK_TRACE "¥nback-tracing¥n"
 88: #  define M_FILE_NOT_FOUND "file not found"
 89: #  define M_FORMAT_ERROR "format error"
 90: #  define M_ILLEGAL_CHARACTER "illegal character"
 91: #  define M_MISSING_PLAYER "missing player"
 92: #  define M_TOO_MANY_PLAYERS "too many players"
 93: #  define M_MISSING_STAR_BLOCK "missing star-block"
 94: #  define M_TOO_MANY_STAR_BLOCKS "too many star-blocks"
 95:
 96: #endif
 97:
 98: /* 手順を覚えるレコードの個数 */
 99: #define MAX_REC ((MAX_BOUND+2)*MAX_TRY)
100:
101: stone map[SIZE_Y + 2][SIZE_X + 2];     /* ステージを表現するマップ */
102:
103: int min_try = MIN_TRY;     /* ステップ数の最小値 */
104: int max_try = MAX_TRY;     /* ステップ数の最大値 */
105: int max_bound = MAX_BOUND;     /* 反射回数の最大値 */
106: int cur_try;     /* 現在何ステップで解こうとしているか */
107: int try;     /* 探索中の現在のステップ数 */
108: int px, py;     /* プレイヤーの位置 */
109:
110: struct {     /* 手順を覚えるレコードの構造 */
111:   stone c;     /* 動いたキャラクタ */
112:   int x1, y1;     /* 動く前の位置 */
113:   int x2, y2;     /* 動いた後の位置 */
114: } rec[MAX_REC];     /* 手順を覚えるレコード */
115:
116: int cnt;     /* 現在使用中のレコード数 */
117:
118: /* 関数の前宣言 */
119: void set_player();
120: void move();
121: void push(int, int, int, int, char [SIZE_Y][SIZE_X]);
122: int move_normal(int, int, int, int, int);
123: int move_left(int, int, int, int, int);
124: int move_right(int, int, int, int, int);
125: int move_star(int, int, int, int, int);
126: void move_player(int, int);
127: int record_move(int, int, int, int);
128: void recur_move();
129: void touch_file(char *, int);
130: int check_star(int, int);
131: void print_map();
132: int load_map(char *);
```

```
133:
134: void exit(int);
135:
136: /* メインルーチン */
137: void main(int argc, char *argv[])
138: {
139:   int t;
140:
141: /* パラメータの読み取り */
142:   switch (argc) {
143:   case 5:
144: /* 反射回数の最大値の指定 */
145:     if (sscanf(argv[4], "%u", &t) == 1 && t <= MAX_BOUND) {
146:       max_bound = t;
147:     } else {
148:       fprintf(stderr, "%s: "M_MAX_BOUND""M_ILLEGAL, argv[0]);
149:       exit(1);
150:     }
151:   case 4:
152: /* ステップ数の最大値の指定 */
153:     if (sscanf(argv[3], "%u", &t) == 1 && t <= MAX_TRY) {
154:       max_try = t;
155:     } else {
156:       fprintf(stderr, "%s: "M_MAX_TRY""M_ILLEGAL, argv[0]);
157:       exit(1);
158:     }
159:   case 3:
160: /* ステップ数の最小値の指定 */
161:     if (sscanf(argv[2], "%u", &t) == 1 && t >= MIN_TRY) {
162:       min_try = t;
163:     } else {
164:       fprintf(stderr, "%s: "M_MIN_TRY""M_ILLEGAL, argv[0]);
165:       exit(1);
166:     }
167:   case 2:
168: /* マップファイルの指定 */
169:     if (load_map(argv[1])) {     /* マップファイルを読み込む */
170:       exit(1);
171:     }
172:     break;
173:   default:
174: /* パラメータが足りないまたは多すぎる */
175:     printf(M_USAGE, argv[0]);     /* 使用法を表示 */
176:     exit(1);
177:   }
178:
179: /* 動作条件を表示 */
180:   printf(M_MAP_FILE": %s¥n", argv[1]);
181:   printf(M_MIN_TRY": %d¥n", min_try);
182:   printf(M_MAX_TRY": %d¥n", max_try);
183:   printf(M_MAX_BOUND": %d¥n", max_bound);
184:   printf("¥n");
185:
186: /* ステージの初期状態を表示 */
187:   printf(M_QUESTION);
188:   print_map();
189:
190: /* プレイヤーの位置を確認 */
191:   set_player();
192: /* レコード番号を初期化 */
193:   cnt = 0;
194:
195: /* メインループ */
196:   for (cur_try = min_try; cur_try <= max_try; cur_try++) {
197:     touch_file("#try.%03d", cur_try);     /* 新しいファイルを作る */
198:     printf(M_TRY_IN, cur_try);     /* 何手で解くか表示 */
199:     try = 1;     /* ステップ数を初期化 */
200:     move();     /* 探索 */
201:   }
202: /* 解けなかった */
203:   printf("¥n");
204:   printf(M_CANT_IN, max_try);
205: }
206:
207: /* プレイヤーの位置を確認 */
208: void set_player()
209: {
210:   int x, y;
211:
212:   for (y = 1; y <= SIZE_Y; y++) {
213:     for (x = 1; x <= SIZE_X; x++) {
214:       if (map[y][x] == PLAYER) {
215:         px = x;
216:         py = y;
217:         break;
218:       }
219:     }
220:   }
221: }
222:
223: /* 探索 */
224: void move()
225: {
226:   char m[SIZE_Y][SIZE_X];     /* プレイヤーが動いた範囲のマップ */
227:   int x, y;
228:
229: /* プレイヤーが動いた範囲のマップを初期化 */
230:   for (y = 0; y < SIZE_Y; y++) {
231:     for (x = 0; x < SIZE_X; x++) {
232:       m[y][x] = 0;
233:     }
234:   }
235:   m[py - 1][px - 1] = 1;     /* 現在位置 */
236:   push(px - 1, py, -1, 0, m);     /* 左に移動or左のブロックを押す */
237:   push(px + 1, py, 1, 0, m);     /* 右に移動or右のブロックを押す */
238:   push(px, py - 1, 0, -1, m);     /* 上に移動or上のブロックを押す */
239:   push(px, py + 1, 0, 1, m);     /* 下に移動or下のブロックを押す */
240: }
241:
242: /* 移動orブロックを押す */
243: void push(int x, int y, int dx, int dy, char m[SIZE_Y][SIZE_X])
244: {
245:   int c0;     /* 押す前のレコード番号 */
246:   int f;     /* ステータス(0=押していない,1=押した,-1=解けた) */
247:
248:   switch (map[y][x]) {
249:   case SPACE:
250: /* 隣がスペースなら移動 */
251:     if (!m[y - 1][x - 1]) {     /* 既に動いた範囲でなければ */
252:       m[y - 1][x - 1] = 1;     /* 動いた範囲をマーク */
253:       push(x - 1, y, -1, 0, m);     /* 左に移動or左のブロックを押す */
254:       push(x + 1, y, 1, 0, m);     /* 右に移動or右のブロックを押す */
255:       push(x, y - 1, 0, -1, m);     /* 上に移動or上のブロックを押す */
256:       push(x, y + 1, 0, 1, m);     /* 下に移動or下のブロックを押す */
257:     }
258:     return;
259:   case NORMAL:
260: /* 隣がノーマルブロックなら押す */
261:     c0 = cnt;
262:     move_player(x - dx, y - dy);     /* プレイヤーの移動を記録 */
263:     f = move_normal(x, y, dx, dy, 0);     /* ノーマルブロックを押す */
264:     break;
265:   case STAR:
266: /* 隣がスターブロックなら押す */
267:     c0 = cnt;
268:     move_player(x - dx, y - dy);     /* プレイヤーの移動を記録 */
269:     f = move_star(x, y, dx, dy, 0);     /* スターブロックを押す */
270:     break;
271:   case LEFT:
272: /* 隣がLブロックなら押す */
273:     c0 = cnt;
274:     move_player(x - dx, y - dy);     /* プレイヤーの移動を記録 */
275:     f = move_left(x, y, dx, dy, 0);     /* Lブロックを押す */
276:     break;
277:   case RIGHT:
278: /* 隣がRブロックなら押す */
279:     c0 = cnt;
280:     move_player(x - dx, y - dy);     /* プレイヤーの移動を記録 */
281:     f = move_right(x, y, dx, dy, 0);     /* Rブロックを押す */
282:     break;
283:   default:
284:     return;
285:   }
286: /* ステータスによる処理 */
287:   if (f > 0) {
288: /* 押したブロックが実際に動いていたら次のステップに進む */
289:     if (try < cur_try) {     /* ステップ数が大きすぎる */
290:       try++;     /* ステップ数を更新 */
291:       move();     /* 探索 */
292:       try--;     /* ステップ数を戻す */
293:     }
294:   } else if (f < 0) {
295: /* 解けたら手順を表示して終了 */
296:     touch_file("#solved.%03d", try);     /* 新しいファイルを作る */
297:     printf(M_SOLVED_IN, try);     /* 何手で解けたか表示 */
298:     printf(M_BACK_TRACE);
299:     print_map();     /* 最終状態のステージを表示 */
300:     while (cnt) {
301:       recur_move();     /* キャラクタを1つ戻す */
302:       print_map();     /* ステージを表示 */
303:     }
304:     exit(0);     /* 終了 */
305:   }
306: /* 移動したキャラクタを元の位置に戻す */
307:   while (cnt > c0) {
308:     recur_move();
309:   }
310: }
311:
312: /* ノーマルブロックの移動 */
313: int move_normal(int bx, int by, int dx, int dy, int n)
314: {
315:   stone c;     /* ぶつかったキャラクタ */
316:   int x = bx, y = by;     /* 移動後のブロックの位置 */
317:
318: /* 反射回数の確認 */
319:   if (n > max_bound) {     /* 反射回数が大きすぎる */
320:     return 0;
321:   }
322: /* 何かにぶつかるまで移動する */
323:   while ((c = map[y += dy][x += dx]) == SPACE) {
324:     ;
325:   }
326: /* ぶつかったキャラクタによる処理 */
327:   switch (c) {
328:   case LEFT:
329: /* Lブロックにぶつかったら左に曲がる */
330:     return record_move(bx, by, x -= dx, y -= dy) |
331:       move_normal(x, y, dy, -dx, n + 1);
332:   case RIGHT:
333: /* Rブロックにぶつかったら右に曲がる */
334:     return record_move(bx, by, x -= dx, y -= dy) |
335:       move_normal(x, y, -dy, dx, n + 1);
336:   }
337: /* その他のキャラクタにぶつかったら止まる */
338:   return record_move(bx, by, x - dx, y - dy);
339: }
340:
341: /* Lブロックの移動 */
342: int move_left(int lx, int ly, int dx, int dy, int n)
343: {
344:   stone c;     /* ぶつかったキャラクタ */
345:   int x = lx, y = ly;     /* 移動後のブロックの位置 */
346:
347: /* 反射回数の確認 */
348:   if (n > max_bound) {     /* 反射回数が大きすぎる */
349:     return 0;
350:   }
351: /* 何かにぶつかるまで移動する */
352:   while ((c = map[y += dy][x += dx]) == SPACE) {
353:     ;
354:   }
355: /* 動くキャラクタにぶつかったらそれを左に移動 */
356:   switch (c) {
357:   case NORMAL:
358: /* ノーマルブロック */
359:     return record_move(lx, ly, x - dx, y - dy) |
360:       move_normal(x, y, dy, -dx, n + 1);
361:   case LEFT:
362: /* Lブロック */
363:     return record_move(lx, ly, x - dx, y - dy) |
364:       move_left(x, y, dy, -dx, n + 1);
365:   case RIGHT:
366: /* Rブロック */
367:     return record_move(lx, ly, x - dx, y - dy) |
368:       move_right(x, y, dy, -dx, n + 1);
369:   case STAR:
370: /* スターブロック */
371:     return record_move(lx, ly, x - dx, y - dy) |
372:       move_star(x, y, dy, -dx, n + 1);
373:   }
374: /* その他のキャラクタにぶつかったら止まる */
375:   return record_move(lx, ly, x - dx, y - dy);
376: }
377:
378: /* Rブロックの移動 */
379: int move_right(int rx, int ry, int dx, int dy, int n)
380: {
381:   stone c;     /* ぶつかったキャラクタ */
382:   int x = rx, y = ry;     /* 移動後のブロックの位置 */
383:
384: /* 反射回数の確認 */
```

▶「ローテク工作実験室」の内蔵AD PCM高音質化計画はよかったです。

小池　克博(23)山梨県

```
385:   if (n > max_bound) {      /* 反射回数が大きすぎる */
386:     return 0;
387:   }
388: /* 何かにぶつかるまで移動する */
389:   while ((c = map[y += dy][x += dx]) == SPACE) {
390:     ;
391:   }
392: /* 動くキャラクタにぶつかったらそれを右に移動 */
393:   switch (c) {
394:   case NORMAL:
395: /* ノーマルブロック */
396:     return record_move(rx, ry, x - dx, y - dy) |
397:       move_normal(x, y, -dy, dx, n + 1);
398:   case LEFT:
399: /* Lブロック */
400:     return record_move(rx, ry, x - dx, y - dy) |
401:       move_left(x, y, -dy, dx, n + 1);
402:   case RIGHT:
403: /* Rブロック */
404:     return record_move(rx, ry, x - dx, y - dy) |
405:       move_right(x, y, -dy, dx, n + 1);
406:   case STAR:
407: /* スターブロック */
408:     return record_move(rx, ry, x - dx, y - dy) |
409:       move_star(x, y, -dy, dx, n + 1);
410:   }
411: /* その他のキャラクタにぶつかったら止まる */
412:   return record_move(rx, ry, x - dx, y - dy);
413: }
414:
415: /* スターブロックの移動 */
416: int move_star(int sx, int sy, int dx, int dy, int n)
417: {
418:   stone c;     /* ぶつかったキャラクタ */
419:   int x = sx, y = sy;     /* 移動後のブロックの位置 */
420:
421: /* 反射回数の確認 */
422:   if (n > max_bound) {     /* 反射回数が大きすぎる */
423:     return check_star(x, y);     /* 3つ並んだかどうか確認 */
424:   }
425: /* 何かにぶつかるまで移動する */
426:   while ((c = map[y += dy][x += dx]) == SPACE) {
427:     ;
428:   }
429: /* ぶつかったキャラクタによる処理 */
430:   switch (c) {
431:   case LEFT:
432: /* Lブロックにぶつかったら左に曲がる */
433:     return record_move(sx, sy, x -= dx, y -= dy) |
434:       move_star(x, y, dy, -dx, n + 1);
435:   case RIGHT:
436: /* Rブロックにぶつかったら左に曲がる */
437:     return record_move(sx, sy, x -= dx, y -= dy) |
438:       move_star(x, y, -dy, dx, n + 1);
439:   }
440: /* その他のキャラクタにぶつかったら止まって3つ並んだかどうか確認 */
441:   return record_move(sx, sy, x -= dx, y -= dy) | check_star(x, y);
442: }
443:
444: /* プレイヤーの移動 */
445: void move_player(int x, int y)
446: {
447:   if (x == px && y == py) {     /* 移動していない */
448:     return;
449:   }
450: /* レコードに記録 */
451:   rec[cnt].c = map[y][x] = PLAYER;     /* 移動したキャラクタ */
452:   map[py][px] = SPACE;     /* 移動前の位置を空白に */
453:   rec[cnt].x1 = px;     /* 移動前の位置 */
454:   rec[cnt].y1 = py;
455:   rec[cnt].x2 = x;     /* 移動後の位置 */
456:   rec[cnt].y2 = y;
457:   cnt++;     /* レコード番号を更新 */
458:   px = x;     /* プレイヤーの位置を変更 */
459:   py = y;
460: }
461:
462: /* ブロックの移動 */
463: int record_move(int x1, int y1, int x2, int y2)
464: {
465:   if (x1 == x2 && y1 == y2) {     /* 移動していない */
466:     return 0;
467:   }
468: /* レコードに記録 */
469:   rec[cnt].c = map[y2][x2] = map[y1][x1];     /* 移動したキャラクタ */
470:   map[y1][x1] = SPACE;     /* 移動前の位置を空白に */
471:   rec[cnt].x1 = x1;     /* 移動前の位置 */
472:   rec[cnt].y1 = y1;
473:   rec[cnt].x2 = x2;     /* 移動後の位置 */
474:   rec[cnt].y2 = y2;
475:   cnt++;     /* レコード番号を更新 */
476:   return 1;
477: }
478:
479: /* 移動したキャラクタを元に戻す */
480: void recur_move()
481: {
482:   int x1, y1;     /* 移動前の位置 */
483:   int x2, y2;     /* 移動後の位置 */
484:
485:   cnt--;     /* レコード番号を戻す */
486:   x1 = rec[cnt].x1;     /* 移動前の位置 */
487:   y1 = rec[cnt].y1;
488:   x2 = rec[cnt].x2;     /* 移動後の位置 */
489:   y2 = rec[cnt].y2;
490:   if ((map[y1][x1] = rec[cnt].c) == PLAYER) {
491:     px = x1;     /* 移動したのがプレイヤーならプレイヤーの位置を変更 */
492:     py = y1;
493:   }
494:   map[y2][x2] = SPACE;     /* 移動後の位置を空白に */
495: }
496:
497: /* 空のファイルを作る */
498: void touch_file(char *s, int n)
499: {
500:   char w[256];
501:   FILE *fp;
502:
503:   sprintf(w, s, n);     /* フォーマットと数値1つでファイル名を作る */
504:   if ((fp = fopen(w, "wt")) != NULL) {
505:     fclose(fp);     /* ファイルがオープンできたときだけクローズ */
506:   }
507: }
508:
509: /* スターブロックが3つ並んだか確認 */
510: int check_star(int x, int y)
511: {
512:   if (map[y][x - 1] == STAR &&
513:     (map[y][x - 2] == STAR || map[y][x + 1] == STAR)) {
514:     return -1;
515:   }
516:   if (map[y][x + 1] == STAR && map[y][x + 2] == STAR) {
517:     return -1;
518:   }
519:   if (map[y - 1][x] == STAR &&
520:     (map[y - 2][x] == STAR || map[y + 1][x] == STAR)) {
521:     return -1;
522:   }
523:   if (map[y + 1][x] == STAR && map[y + 2][x] == STAR) {
524:     return -1;
525:   }
526:   return 0;
527: }
528:
529: /* ステージを表示 */
530: void print_map(int k)
531: {
532:   int x, y;
533:
534:   for (y = 0; y < SIZE_Y + 2; y++) {
535:     for (x = 0; x < SIZE_X + 2; x++) {
536:       PRINTF_STONE(map[y][x]);
537:     }
538:     printf("¥n");
539:   }
540:   printf("¥n");
541: }
542:
543: /* マップファイルを読み込む */
544: int load_map(char *fn)
545: {
546:   FILE *fp;     /* ファイルハンドル */
547:   int x, y;     /* 座標 */
548:   int kp = 0;     /* プレイヤーの数 */
549:   int ks = 0;     /* スターブロックの数 */
550:   int kf = 0;     /* 読み込みエラーがあったか */
551:   int kc = 0;     /* キャラクタの間違いがあったか */
552:   int kx = 0, ky = 0;     /* 間違っている位置 */
553:   stone c;     /* 読み込んだキャラクタ */
554:   char ch;     /* 改行を読み込むためのワーク */
555:
556: /* ファイルをオープンする */
557:   if ((fp = fopen(fn, "rt")) == NULL) {
558:     fprintf(stderr, "load_map: "M_FILE_NOT_FOUND" (%s)¥n", fn);
559:     return 1;
560:   }
561: /* すべての座標についてループ */
562:   for (y = 0; y < SIZE_Y + 2; y++) {
563:     for (x = 0; x < SIZE_X + 2; x++) {
564:       if (feof(fp)) {     /* 予期しないファイルエンド */
565:         kf = 1;
566:         break;
567:       }
568:       if (FSCANF_STONE(fp, c) != 1) {
569:         kf = 1;     /* 所定のフォーマットでキャラクタを読み込めない */
570:         break;
571:       }
572: /* 読み込んだキャラクタの簡単なチェック */
573:       switch (c) {
574:       case STAR:     /* スターブロック */
575:         ks++;
576:       case PLAYER:     /* プレイヤー */
577:         kp++;
578:       case SPACE:     /* 空白 */
579:       case NORMAL:     /* ノーマルブロック */
580:       case LEFT:     /* Lブロック */
581:       case RIGHT:     /* Rブロック */
582:         if (x == 0 || x == SIZE_X + 1 || y == 0 || y ==  SIZE_Y + 1
) {
583:           if (!kc) {
584:             kx = x;     /* 壁か固定ブロックでない */
585:             ky = y;
586:             kc = 1;
587:           }
588:         }
589:       case WALL:     /* 固定ブロック */
590:         break;
591:       default:
592:         if (!kc) {
593:           kx = x;     /* 間違ったキャラクタがある */
594:           ky = y;
595:           kc = 1;
596:         }
597:       }
598:       map[y][x] = c;     /* キャラクタをマップにセットする */
599:     }
600:     if (fscanf(fp, "%1c", &ch) != 1 || ch != '¥n') {
601:       kf = 1;     /* 改行を読み込めない */
602:     }
603:     if (kf) {
604:       break;
605:     }
606:   }
607: /* クローズ */
608:   fclose(fp);
609:
610: /* エラーメッセージを表示 */
611:   if (kf) {     /* 読み込みエラー */
612:     fprintf(stderr, "load_map: "M_FORMAT_ERROR"¥n");
613:   }
614:   if (kc) {     /* キャラクタの間違いは位置を添えて報告 */
615:     fprintf(stderr, "load_map: "M_ILLEGAL_CHARACTER" (%d,%d)¥n", kx
, ky);
616:     kf = 1;
617:   }
618:   kp -= ks;
619:   if (kp < 1) {     /* プレイヤーが指定されていない */
620:     fprintf(stderr, "load_map: "M_MISSING_PLAYER"¥n");
621:     kf = 1;
622:   } else if (kp > 1) {     /* プレイヤーが複数指定された */
623:     fprintf(stderr, "load_map: "M_TOO_MANY_PLAYERS"¥n");
624:     kf = 1;
625:   }
626:   if (ks < 3) {     /* スターブロックが足りない */
627:     fprintf(stderr, "load_map: "M_MISSING_STAR_BLOCK"¥n");
628:     kf = 1;
629:   } else if (ks > 3) {     /* スターブロックが多すぎる */
630:     fprintf(stderr, "load_map: "M_TOO_MANY_STAR_BLOCKS"¥n");
631:     kf = 1;
632:   }
633:
634:   return kf;
635: }
```

▶読みものが好きです。特に「知能機械概論」は毎回楽しみにしています。

桜井　誠(24)埼玉県

ダイアログもどきの作成

Ishigami Tatsuya 石上 達也

今回はSX-BASICのプログラムの動作の様子とdi( )関数による高速化の実際について解説します。また，SX-BASICによる疑似ダイアログ処理のようなものについても検討してみましょう。

中間コードの実行

SX-WINDOWは複数のプログラムが同時に動作するマルチタスクシステムです。SX-BASICはSX-WINDOW上で動作するプログラムですから，当然，ほかのプログラムとも同時に動作します。

ここで，SX-WINDOW上でBASICと2種類のプログラム（それぞれプログラムA，プログラムBとします）が実行されていたとします。

たとえば，

```
10 int a,b
20 b = 1+2
```

というプログラムはSX-BASICによって中間コードへと変換されます。ここまではまったく問題はありません。そして中間コードの実行へと移ります。中間コードとは，

　「1」をある場所にセット
　「2」をある場所にセット
　直前にセットされた値とさらにその前にセットされた値との和を2番目に宣言された*int*型変数に代入
　終了

というようにBASICとマシン語の中間くらいの抽象度を持ったコードデータです。このデータを逐次，実行していけば目的は果たせるわけですが，SX-WINDOWはマルチタスク環境ですから，実行順序は，

　「1」をある場所にセット
　～タスクAをちょっと実行～
　～タスクBをちょっと実行～
　～タスクAをちょっと実行～
　～タスクBをちょっと実行～
　直前にセットされた値とさらにその前にセットされた値との和を2番目に宣言されたint型変数に代入
　～タスクAをちょっと実行～
　～タスクBをちょっと実行～
　終了

ということになります（先月号でも少し触れましたが，これらはSX-WINDOWのヌルイベント時に行われます）。

diとei

パソコン本体にはCPUがひとつしかないくせに，SX-WINDOWはマルチタスク環境を実現しなければいけないので，前述のようにお互いのプログラムが譲りあって動作していたのです。

プログラムが譲りあっていれば，すべてがうまくいくかというとそうでもありません。

C言語やアセンブラで書かれたプログラムは，明示的に譲ってもよい場所と譲れない場所というものが示されています。しかし，前述のようにSX-BASICでは中間コードひとつ実行するたびに，無条件に譲りあっていますので，譲りあってはいけないような機能を実現するのには不向きです。

SX-WINDOWには「カレント～」という概念が多く出てくるのですが，Currentとはこの場合「現在の～」というくらいの意味です。「カレントドライブ」とは「現在いじっているドライブ」のことですし，「カレントディレクトリ」というのは「現在いじっているディレクトリ」ということでした（詳細は「Human68kマニュアル」を参照）。あれと同じ意味です。

SX-WINDOW上のプログラムは自分で開いたウィンドウ上にしか描画を行えないことになっていますが，これは「カレントグラフ（Current Graph）」を自分の開いたウィンドウへとセットしてから描画を行うべしという約束の下に実現されています。

また，いろいろ話が複雑になってしまうのですが，ほかのプログラムにメモリコンパクションされては困るような場合もあるかもしれません。

実行する際に，ほかのタスクと譲りあわれては困るような部分にはdi()関数を事前に実行します。もちろん，SX-WINDOW環境はすべてのプログラムの譲りあいの精神のうえに成り立っていますので，譲りあわない部分は必要最小限にとどめておかなければなりません。

譲りあいを再開するには，ei()関数を実行します。例によって，

```
10 int a,b
20 di()
30 b = 1+2
40 ei()
40 print b
```

というプログラムは，

　譲りあいを禁止
　「1」をある場所にセット
　×～タスクAをちょっと実行～
　×～タスクBをちょっと実行～
　「2」をある場所にセット
　×～タスクAをちょっと実行～
　×～タスクBをちょっと実行～
　直前にセットされた値とさらにその前にセットされた値との和を2番目に宣言されたint型変数に代入。
　×～タスクAをちょっと実行～
　×～タスクBをちょっと実行～
　譲りあい再開
　～タスクAをちょっと実行～
　～タスクBをちょっと実行～

2番目に宣言されたint型変数の値を画面に10進数で表示

～タスクAをちょっと実行～

～タスクBをちょっと実行～

終了

という具合になります。わかる人にはわかると思いますが，

di : disable interrupt

ei : enable interrupt

の略です。

疑似ダイアログ

データを生成したり，編集したりするプログラムの場合，ユーザーにファイル名の入力を促したい場合があります。

1) まず，手軽に実現できる方法として，ウィンドウ中にファイル名を表示するテキストアイテムを配置する，というのが考えられます。このテキストアイテムを編集可能にしておけば（editableプロパティ＝1），それをユーザーが書き換えることによって，プログラムはファイル名を得ることができます（写真1：リソースゲッタ「開発キットツール集」収録）。

2) また，これとは別にファイル名入力を促すダイアログや専用のウィンドウを開く方法があります（写真2：ウィンドウデザイナ）。

3) そして，最近多くみられる「疑似ダイアログ」方式（写真3：シャーペン）。

と，およそ3種類の方法が考えられます。

1)は，配置したテキストアイテムのKeyInイベントを待ち，captionプロパティを参照するだけですから，特に問題はありません。各自でトライしてください。

2)は，現在のところ，SX-BASICが複数のウィンドウ操作を行えませんのでかなりトリッキーな実現方法となります（おそらく，ファイル名入力用のウィンドウを司るSX-BASICのプログラムをfock関数で呼び出すことにより可能と思われますが，実際はわかりません）。

先月号の楽譜エディタではinputbox$関数を用いることにより，なんとかそれに近いことを実現していましたが(写真4)，ファイル出力/入力に関するキャンセルが効かないという欠点を持っていました（ボタンがひとつしかないので，常に「確認」の意味にとられてしまう）。

この件に関しては複数ウィンドウ，複数ページのサポートという形での解決を考えています。

さて，残った3)の方法を考えてみましょう。

「疑似ダイアログ」とは，ダイアログのようでいて，実はダイアログではありません。ただ，ウィンドウの中に写真2のような模様が描かれ，そこにテキストアイテムと標準ボタンが2つ置かれているだけです。

ファイル入出力操作のときのみ出現し，それ以外のときにはじっとなりを潜めています。

このように，出現させたり，隠したりするときにはvisibleプロパティを使います。

写真2では，ダイアログ部分の四隅がネジで止められていたりしてそれなりの雰囲気を醸し出しているのですが，まず，初めはこの部分を質実剛健に，height＝1のレク

写真1 リソースゲッタ

写真2 ウィンドウデザイナ

写真3 シャーペン

リスト1

```
 1: ▼Window Size (240,171),0,0,疑似ダイアログ（その1）
 2: /* ここで、初期化に必要な処理を行なって下さい
 3: CloseDialog() /*疑似ダイアログを消去する
 4: end
 5: func File_Drop(filename;str)
 6: endfunc
 7: ▼3,StnBtn1 (56,136,200,156),0,0,疑似ダイアログを表示
 8: func StnBtn1_Click()
 9:   OpenDialog()
10: endfunc
11: /* 疑似ダイアログを開く
12: func OpenDialog()
13: di()
14:   Rect1.visible = 1
15:   StnBtn2.visible = 1
16:   StnBtn3.visible = 1
17:   Text1.visible = 1
18:  Text1.active
19:   Text2.visible = 1
20: ei()
21: endfunc
22: ▼2,Rect1 (12,12,204,96),0,0,1
23: ▼1,Text1 (28,40,176,60),0,0,0,0,3,1,0,1,
24: func Text1_KeyIn(s;str)
25:   StnBtn2_Click() /*リターンキーが入力されたのは「設定」ボタンが押されたのと同じ
26: endfunc
27: ▼1,Text2 (20,24,184,40),0,0,0,1,3,0,0,0,ファイル名を入力して下さい
28: func Text2_Click()
29: endfunc
30: ▼3,StnBtn2 (100,68,136,88),0,0,設定
31: func StnBtn2_Click()
32:   CloseDialog()
33:   Text3.caption = Text1.caption
34: endfunc
35: /* 疑似ダイアログを消去する
36: func CloseDialog()
37:   Rect1.visible = 0
38:   StnBtn2.visible = 0
39:   StnBtn3.visible = 0
40:   Text1.visible = 0
41:   Text2.visible = 0
42: endfunc
43: ▼3,StnBtn3 (140,68,176,88),0,0,取消
44: func StnBtn3_Click()
45:   CloseDialog()
46: endfunc
47: ▼1,Text3 (32,108,240,132),0,0,0,1,3,0,0,0,ファイルは選択されていません
48: func Text3_Click()
49: endfunc
```

写真4 疑似ダイアログもどき作成の過程

写真5 その実行時

タングルアイテムに置き換えて実現してみます。ウィンドウデザイナにより，写真4.1〜4.8のようなウィンドウを作ってください。リスト1にその際に生成されるファイルを示します。

ウィンドウのタイトル部分から，右プレスによりメニューを表示させ「File Load」か「File Save」を選んでください。するとファイル名入力用の疑似ダイアログもどき(!)が表示されますので，適当な文字列を与え，リターンキーを押すか，「確認」「取消」いずれかのボタンを左クリックしてください。

「取消」を左クリックした場合を除き，ウィンドウ中央部分に与えたファイル名が表示されているはずです。

ただし，このプログラムはファイル名を取り込むためのサンプルですので，実際にファイルの読み書きを行うわけではありません。

autoexec(-r1オプションとの違い)

SX-WINDOWには，Human68kで使われていたコマンドライン機能が残されています。プログラム起動時に引数を与える機能です。

実行ファイルのアイコンをダブルクリックすると，普段はそのプログラムが起動されます。しかし，ダブルクリックする際に，キーボードのOPT.1キーを押していると，「コマンド起動」のウィンドウが開きます。このウィンドウは文字列の入力を促してくるのですが，ここで入力する文字列は，ほとんどHuman68kで与えるコマンドラインと同じものです。

OPT.1キーを押さなくても「アイコンメンテ」を用いて自動的にコマンドラインを指定することもできます。

現在，SX-BASICのプログラムは拡張子「.sxb」ということになっていると思いますが，このようなファイルがダブルクリックされた場合，ユーザーの意図は2通り考えられるわけです。

1) まだプログラムが完成していないので，その続きを行いたい。

2) そのファイルには完成されたプログラムが入っているので，面倒なことはいわずにすぐ実行を始めたい。

1)は，「アイコンメンテ」で実行オプションを「-f%」に，2)は「-f% -r1」に指定します。

さて，これとは別にSX-BASICにはデスクトップ画面を保存する機能があります。作業の途中でSX-WINDOWを終了すると次回の起動時にはその状態が保存されているという機能です。

前回，読み込まれていたファイルがあればSX-BASICもそれを読み込むようになっています。で，ファイルをSX-BASICが読み込む場合，2通りの意味あいがあるわけです。

1) 前回のSX-WINDOW終了時，プログラムの入力途中だった。

2) 前回のSX-WINDOW終了時，プログラムの実行途中だった。

この2つの状況を自動的に判別すればよいかというと，そうもいかず，たとえば，以下のような例もあるわけです。

1) プログラムを入力後，実行させたが，

ちょっと悪いプログラム

SX-BASICはpeek( )/poke( )できるので，最近直接コントローラに値を叩き込んだりシステムポートいじったりと悪のプログラム作りに励んでいます。

たとえば，いくらウィンドウ環境で隣にディレクトリウィンドウ出して作業できるからって，ファイルの読み込みとかを全部それにまかせるというのはイマイチ使いづらいですよね。個々のプログラムでもファイルセレクタくらいは使えたほうがいいと思いませんか？

しかしSX-BASICにはそんな機能はありません。んで，タスクマンの情報を取ってくるわけですが，ここでどうしてもメモリマンで非リロケータブルブロックを確保する必要があります。これはちょっとややこしいんですが，SX-BASICでやると右のリストのようになります。ポインタったって，ただの整数で大丈夫。

この例では単にファイル名を表示するだけです。終了時には確保したメモリを解放してやらないとメモリにゴミが残ってもったいないので気をつけてください。 (S.N.)

```
 1: /*
 2: /* ディレクトリ表示のテスト
 3: /*
 4: /*
 5: /*                              S.NAKANO
 6:
 7: int a(10)
 8: str d,f,g
 9: int b,c,e,i,j,k,l,m,n
10: int buf(90)
11:
12: d="d:¥picture¥*.pic"
13: m=getmem(16)
14: n=getmem(360)
15: asets(m,d)
16: print
17: print A_line(&ha370,32;w,m;l,n;l)
18:                     /* FILES
19: i=n+30
20:
21: repeat
22:     j=peek(i)
23:     print chr$(j);
24:     i=i+1
25: until j=0
26: repeat
27:     e=nfiles(m,n)
28: until e<0
29:
30: end
31:
32: func asets(k,d;str)
33: int l,s,t,u,v,z
34: z=1
35: l=len(d)-1
36: for s=0 to l/4
37:     for t=0 to 3
38:         u=asc(mid$(d,z,1))
39:         if z>l+1 then u=32
40:         poke(k+z-1,u)
41:         print chr$(peek(k+z-1));
42:         v=u+(v shl 8)
43:         z=z+1
44:     next
45:     buf(s)=v
46: next
47: endfunc
48:
49: func getmem(q)
50: return(A_line(&ha01e,q;l))
51: endfunc
52:
53: func nfiles(m,n)
54: int e,i,j
55: e= A_line(&ha371,32;w,m;l,n;l)
56:         /* NFILES
57: i=n+30
58: if e>=0 then {
59:     repeat
60:         j=peek(i)
61:         print chr$(j);
62:         i=i+1
63:     until j=0
64: print
65: return(e)
66: endfunc
```

そのプログラムに（あるいはSX-BASIC本体に）不具合があり，やむなくSX-WINDOWを「終了」した。
2) SX-WINDOWを再起動したが，SX-BASICは問題のあるプログラムの実行をいきなり開始してしまった。
3) デバッグするどころかSX-WINDOWを再起動できなくなり，しかたなくsysdtop.sx，bootdtop.sxをマスターディスクからコピーしてきた。

このような状態に陥るのは，SX-BASICがまだまだ不安定な状態にあるからで（かっこわりー），ゆくゆくは，この問題は解決されることでしょう。

で，自動的には行えませんが，手動でならば行えるようになっています。SX-BASICは起動時にファイル指定があった場合，「-r1」オプションが指定されているかどうかで，即時実行を行うか，それとも，「run」と入力されるのを待つようにするか決めます。ですから，そのどちらを希望するのかは「-r1」オプションをつけるかどうかにかかっています。

さて，このコマンドラインですが，タスクモニタなどがあれば変更を加えることができるのですが，あまりスマートな方法ではありません。SX-BASICではautoexecを使うことにより，このコマンドオプションに変更を加えることができるようになっています。autoexecは入力されたSX-BASIC自身のコマンドオプションを書き換えます。それだけの操作です。

ですから，「アイコンメンテ」などで「-r1」オプションが指定されていた場合は結局同じことですからautoexecは気にしなくてもかまいません。

モニタ部分の扱い

SX-BASICコンパイラ発表後，モニタ部分を廃止するかもしれない，というようなことを何回か書きました。

現在，モニタ部分には，プログラム入力，文字列の入出力（print文やinput文）という2つの役割があります。

SX-BASICでは，行番号エディタという一昔前のエディタを使用しています。これが嫌ならシャーペンなどのエディタを用いればよいわけです。テキスト文字だけでプログラムを扱うには問題はありません。

いま手元にある最新バージョンでは，試験的にマルチフォントテキストのサポートを行っています。変数への代入などは行えませんが，クリップボードやテキストアイテムへの代入なら，なんとかできるようになっています（おそらく，10月号の付録ディスクでも状況は変わらないはずです。すみません）。

もしこのアプローチが有効であるとしたら，変数の型や組み込み関数を大幅に追加し，本格的にマルチフォントテキストをサポートする予定ですが，その場合，ラインエディタでは，やや役不足です。

SX-WINDOW ver.3.1ではシャーペンの外部コマンドがサポートされているようですので，SX-BASICの外部コマンド化という方向で現在検討中です。

仮にラインエディタが廃止されても，ダイレクトコマンドの実行やprint文やinput文での用途は残るわけです。これらの機能を使った対話性というのはBASICの大きな特徴ですから，ぜひともこの機能は残しておこうと思っています。

しかし，SX-BASICのプログラムがコンパイルされた場合を考えてください。最終的には，*.x形式の実行ファイルになります(その予定です)。ひとつの完成されたプログラムが，自身のウィンドウ（インタプリタ状態でのウィンドウエンジン）のほかにもウィンドウ（print文やinput文が使用）を開くというのは，あまり自然な状態ではありません。

一応，コンパイラスイッチかなにかで，選択するようなことも考えていますが，場合によると，コンパイラはprint文やinput文をサポートしない可能性もあります。

投稿について

編集部には，SX-BASICを使用した投稿が寄せられ始めています。私がいうのも変な話ですが，なかには「へぇー，SX-BASICってこんなこともできるのか」と驚いてしまうような力作もあります。私もこれらのプログラムが誌面上で発表されるのを楽しみに待っています。完成度は文句ないのに，ただただプログラムの大きさの問題で掲載を見送られていたものもありますが，10月号の付録ディスクでは，このうちの何本かは収録されると思います。

編集部に投稿していただいたということは掲載されることを希望して送ってきたということだと思うのですが，Oh!Xにはプログラム容量のほかにも，もうひとつの制限があります。Oh!Xは定価がついて本屋で売られている商業誌です。個人的に楽しむ範囲を越えています。

というわけで，やっぱりセーラームーンはちょっとまずいのです。というわけで，セーラームーン以外の投稿をお願い致します（編注：投稿にはセーラームーンではないと書いてありました。しかしセーラー服はともかく，やはりティアラはマズいのではないかと……）。

予告

5月号の「こいのぼりPRO-68K」では，ファイル容量の関係から，ソースリストを添付することができませんでした。パソコン通信に参加しているスタッフに協力してもらいソースリストをアップロードしてもらったので，これをダウンロードするか，編集部に配布を申し込むかの2種類の形で，ソースリストを入手していただかなくてはなりませんでした。

前者の場合は特に問題ないのですが，後者の発送作業がなかなか進んでいません。

現在，マルチフォントテキストのサポート，フロートウィンドウ，メニューバーのサポートなどSX-BASICに大幅な改造を加えています。順調に作業が進めば問題ないのですが，本体の基幹となるような部分にまで改造を加えているので，あっちを立てればこっちが立たずというように一進一退を繰り返しています(とほほ)。さらに悪いことに，なかなか，ソースリストが配布できるような状況になりません。

編集部内からは，10月号は付録ディスクか？ というような声も聞こえてきます（あわわ)。下手をするとソースリストの形式で配布するバージョンが，そのまま付録ディスクに収録されるバージョンだったという事態も考えられなくもありません。せっかく申し込んでくださってたいへん申しわけないのですが，そのような事態になったらごめんなさい。

CGA入門キット「GENIE」(その3)

プロジェクトチームDōGA
かまた ゆたか

少しだけ残っていた「GENIE」の応用編を解説します。そのあと，最近たまりにたまっている質問のお手紙に，4月号のアンケート結果と併せて，答えていきましょう。

はじめに

実際に原稿を書いているときと，それが雑誌になって，さらにその反響が出るまでには，時間的にかなり間が空きます。こちらには，いまごろやっと7月号の「GENIE」の反響のお手紙が届いているところです。狙いどおり，CGA初心者や，途中で挫折していた方には好評なようで，苦労して制作したかいがあったというものです。

とはいっても，さすがの「GENIE」も3カ月目になると，もう飽きてしまっているでしょう。しかし，いまからCGAシステムのマニュアルを申し込んでも，手に入るのは10月です。また，いまは何もしていなくても，自分で作品を作るようになったときや，何か急にデモのCGなんかが必要になったとき，もう一度この「GENIE」を取り出して，メカデザインすることもあるでしょう。

ということで，この「GENIE」をいろいろ応用するために，まず中身を解析するところから始めましょう。

ディレクトリの解析

以下，ハードディスクのシステムを中心に解説します。「GENIE」をインストールしたディレクトリを見ると，genie.batのほかに，7つのディレクトリがあります(写真1)。それぞれ，何が入っているのでしょうか。

1) BIN

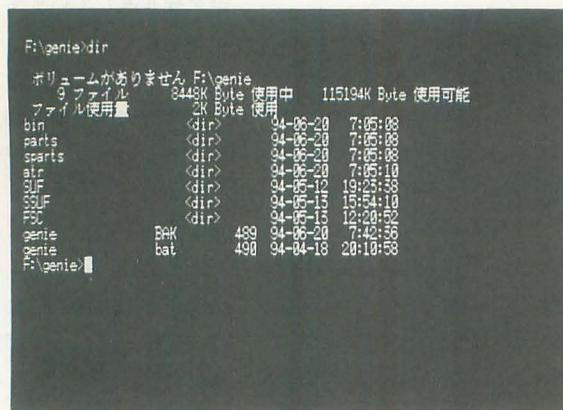



写真1　GENIEのディレクトリ構成

まず「bin」には，CGAシステムのプログラムが入っています。また，「*.WIN」や「*.K」は，「GENIE」のメニューシステム関連のファイルです。

「FF.X」「FFE.X」「HANIM.X」「KAMA.X」「REND*.X」「WIREVIEW.X」といったツールは，たぶん皆さんがお持ちのCGAシステムのものよりバージョンが新しいと思います。CGAシステムのディレクトリにコピーして使ってもよいでしょう。ただし，大きなエンバグをしている可能性がありますので，古いバージョンのツールも「*.OLD」にリネームして残しておいてください。具体的にどのようにバージョンアップしたかについては，現在申し込みを受けつけているバージョンアップ追加マニュアルのほうに詳しく掲載する予定です。

「GENIE」しか持っていない方は，コマンドラインからこれらのツールを使用する場合，この「bin」にパスを通しておく必要があります。たとえば，A:のルートに「genie」というディレクトリを作って，そのなかに「GENIE」を入れていたとすると，

path a:¥genie¥bin;%path%

とします。

2) PARTS

このディレクトリは，「戦闘機」「翼」「エンジン」といった，パーツの形状のデータベースです。「GENIE」で「物体追加」を行うときに見えているのがこのディレクトリです。

3) SPARTS

一見したところ，「parts」のディレクトリとまったく同じように見えます。「戦闘機」といったディレクトリがあり，そのなかにはちゃんと「FT01.SUF」から「FT09.SUF」まであります。しかし，同じディレクトリ名，同じファイル名だからといって，内容まで同じわけではありません。同じパーツのシンプル形状(形は似ているが面数をきわめて省略した形状)が入っています。

「GENIE」でメカデザインするとき，たとえば「戦闘機」の「FT03.SUF」に「翼」の「WG12.SUF」をつけるといった操作を行いますが，このシンプル版のディレクトリでその操作を行うと，モーションデザインのときに使うシンプル版のメカができるというわけです。

4) ATR

色に関する情報のアトリビュートファイルが6つ入っています。「GENIE.ATR」がデフォルトの色で，「デザインしたメカを確認する」のときはこの色になります。

あとはご想像どおり，メカの色調変更用のアトリビュートで，「GENIE_WH.ATR」が白，「GENIE_BL.ATR」は青，「GENIE_RD.ATR」は赤，「GENIE_GR.ATR」が緑，そして「GENIE_MG.ATR」が紫です。

ですから，「青の色調を選択すると，暗すぎて見えにくい」とか「紫なんて使わないけど黄色い色調が欲しい」なんてときに，このへんのファイルの内容を変更すればよいのです。アトリビュートの各パラメータに関する解説は，CGAシステムのマニュアルをご覧ください。

5) SUF

自分でデザインしたメカが入ります。最初は，サンプルデータだけが入っています。

たとえば「OHX01」というメカを作ると，「OHX01.FSC」と「OHX01.SUF」ができます。「OHX01.FSC」は，どのパーツをどのようにくっつけたかという設計図に当たる情報が入っています。ちょっとした変更なら，エディタで直接このファイルを書き換えてやることもできるでしょう。「OHX01.SUF」は，この設計図を基に，KAMA.Xでパーツを組み合わせた形状です。「GENIE」では，「新たにメカをデザインする」や「前にデザインしたメカを変更する」の処理を行ったのち，タイムスタンプが新しい*.FSCがあると，自動的にKAMA.Xを実行するようになっています。

また，「デザインしたメカを確認する」を実行すると，新たに「OHX01」というディレクトリが作られます。ここには，「デザインしたメカを確認する」で作画させた画像などが入っています。ただ，FD版では，ディスク容量に制限があるため，表示ののち自動的に削除します。ですから，再度実行したとき，HD版ではすぐ画像が表示されるのに対し，FD版はいちいち作画し直すわけです。

6) SSUF

このディレクトリには，上記の「*.FSC」という設計図に従って，「sparts」のディレクトリのなかのシンプル版のパーツを組み合わせて作られたシンプル版の形状が入っています。ファイル名がまったく同じなので，こうして別のディレクトリに入れておくわけです。

7) FSC

ここには，「動きをデザインする」で作られた動きのデータ「*.fsc」や，「アニメーションを作る」で作られた画像データなどが入っています。初めは，サンプルの動き「SAMP?.FSC」しか入っていません。

たとえば，「TEST1」という動きを設定し，作画・アニメーションさせた場合，「TEST1」というディレクトリが作られ，そのなかに「TEST1*.PIC」という画像データが作られます。

いらないデータの削除

「GENIE」で遊んでいると，オリジナルメカやアニメーションのデータが増えて，メニューが多くなり，表示するのにもいちいち時間がかかるようになってきます。そうなったら，不要なデータは消してしまいましょう。

本来，不要なデータを削除する機能ぐらい，「GENIE」が持っているべきですが，そのことに気がついたのがマスターアップの直前で，エンバグがこわいので，もうそのままにしてしまいました。ごめんなさい。

1) オリジナルメカの削除

「BOTU」というメカを削除する場合，前述のディレクトリの解説を読めばだいたい予想がつくように，「SUF」というディレクトリのなかの「BOTU.SUF」と「BOTU.FSC」を削除します。

```
DEL BOTU.*
```

で一挙に消せます。また，「BOTU」という名のディレクトリができている場合は，

```
DEL BOTU
RD  BOTU
```

も必要です。

さらに，シンプル版の形状も不要ですので，「SSUF」のなかの「BOTU.SUF」も削除します。

2) 動き，アニメーションの削除

同様に，「FUYOU」という動きを削除するときは，「FSC」のなかの「FUYOU.FSC」を消します。

その動きを作画させたことがある場合，「FUYOU.FRM」や「FUYOU.TCH」もできているので，これも消します。さらに「FUYOU」というディレクトリもあるので，中身を消して，このディレクトリも削除します。

オリジナルパーツを加える

7月号のカラーのページの一覧表を見て，すごいデータ量だと思った人も，自分でメカを作り始めると「こんな翼が欲しい」とか「パラボラアンテナがない」など不満が出てくるでしょう。残念ながら，いまのところ追加データ集を出すというウワサはないので自分で作るしかありません。もう一度，あのCAD.Xと格闘してください。

まず，パーツをモデリングするときの問題としては，アトリビュート名に注意してください。でたらめな名前をつけても，ちゃんと作画してくれません。表1のなかから選択してください。

大きさや向きは，くっつけ

表1　アトリビュート名

BodyM	本体のメインの金属
BodyL	本体のメインより少し明るい塗装
BodyD	本体のメインより少し暗い金属
WinLg	艦橋の窓などの白い明かり
EngLg	噴射口の少し青い明かり
Glass	コックピットの青いガラス
GrayG	ゴムのような暗い灰色
RedLg	マークなどの赤い明かり
YelLg	マークなどの黄色い明かり
Black	ほとんど真っ黒
Brown	ノズルなどの錆びついた金属

るときに調整できますが，基本はCAD.Xの起動時のスケールで画面内に余裕を持って収まる程度の大きさで，X軸の＋方向が前，Y軸の＋方向が左で作るとよいでしょう。

また，新たにパーツを加える場合，そのパーツのシンプル版も作らなければいけません。シンプル版は，非常に大ざっぱな形で，面数は10面以下で十分です。もともとの形状の面を削っていくのではなく，新たにアウトラインをなぞるような大きな面を，XY平面，YZ平面，ZX平面に1面ずつ作るのが基本です。

シンプル版の形状も，同じファイル名(オブジェクト

Q&A特集

今年のCGAコンテストのビデオの申し込みの際，通信欄にてさまざまなご意見，ご質問や励ましのお便りをいただきました。今回は，4月号のアンケートの集計結果やコラム「500円の攻防」についてのご意見を中心に，皆さんからのご質問に答えていきたいと思います。積極的な意見が多く，こうしてまとめてみると，誌上討論会みたいです。

＊　＊　＊

「500円の攻防」とは，コンテストのビデオの価格をどうするかという論争でした。アンケートを集計すると，以下のようになりました。

A) 2,500円：ビデオの実費以外のコンテスト運営費のカンパを募るのは不当だ［2％］
B) 3,000円：コンテスト運営費の一部を補填する程度のカンパはよい［54％］
C) 3,500円：コンテストの開催全体の採算がとれる健全な運営を行うべきだ［41％］

ご覧のように，B)とC)に二分されています。ところで，集計していて気がついたのですが，早い時期に申し込まれた方(コンテストに対して積極的な方？)はほとんどC)でしたが，最後のほうに申し込まれた方はB)が圧倒的に多くなっていました。

B)やC)の方の一般的な意見は次のような感じでした。

「コンテストが赤字で運営できなくなるほうが大きな損失だから，多少の値上げはかまわない」
「コンテストを運営するうえで，経費が発生するのであれば，当然その収入源を確保すべき。ビデオ販売でもかまわないと思う」

ご理解あるご意見ありがとうございます。

次に，少数派ですが，A)と答えた方の意見も見てみましょう。

藤岡市Mさん**「カンパは強制するものであってはならないし，かといってカンパがなければコンテストの運営が危うい。でもコンテストの費用をビデオ代に加えるのは絶対におかしい」**

といいつつも，この方は1,000円カンパして，3,500円送っていらっしゃいます。つまり，カンパは強制されなくても，ちゃんと出すということでしょうか。

水戸市Aさん**「わけのわからない『カンパ』より，『コンテスト運営金』として500円を支払うほうがすっきりします」**

確かに一理あるご意見です。

また，これもごく少数派ですが，こういうご意見もあります。

宝塚市Nさん**「あなた方には，儲けてもらいたい。実費＋利益で算出して何が悪い。むしろカンパは変だ。一般客を相手にしてんだぞ……」**

う～ん，一般客を相手にしているつもりはありません。DōGAの活動は，単なる営利活動ではなく「CGAを通じて，日本にパーソナル映像文化を育てよう」という運動です。ですから，企業と客という関係ではなく，同じ目的を持った仲間だと思っています。それで，同じ仲間から実費以上の利益を得ることに抵抗を感じるわけです。まぁ，理想論かもしれませんが。

また，不満としては，次の意見が非常に多くありました。

「赤字の根源であるコンテストの上映会が，なぜ入場無料なのか。ビデオと同様カンパを募ることはできなかったのか。上映会に行っていない人(ビデオ申し込み者)が，上映会の赤字を負担するのはおかしいではないか。500円程度の入場料を取るべきだ」

一応，会場内にカンパの箱(御賽銭箱)は設置しています。しかし，コンテストの出費のうち，上映会が占める割合はかなり高く，それをビデオの申し込み者が負担するのは確かに理不尽かもしれません。

でも，入場料を取ると問題も多く発生します。まず，CGAコンテストはアマチュアの祭典であり，観客が「俺たちは金を払って見てやっているお客だぞ」という雰囲気にはしたくないということです。また，現実的に考えて，700人の来場者から500円ずつ集めても35万円にしかなりません。これはコンテスト開催費全体の10分の1程度です。さらに，2,500円のビデオが3,000円になるのと，無料の入場料が500円になるのとでは影響力が違います。

ビデオ配布が主に固定ファンに対するサービスなのに対して，上映会には，新規ファンの開拓という意義があります。会場のアンケートには「あまり興味がなかったが，友人に誘われて来て，非常に面白かった」というのがよくありますが，このような人たちは入場料があると来にくいでしょう。

ということで，確かに入場料を取らないのは理不尽だけれども，取ったところで，メリットはほとんどなく，デメリットが大きいといえます。上映会は，上映を見に来る人のためだけに行われているわけではありません。応募者同士の交流，コンテスト自体のPRなど，コンテストの主旨からいっても必要なものであり，間接的にビデオの応募者にも関係ある問題です。ご理解いただけますでしょうか。

さて，コンテストの具体的な財源としては，以下の提案が多くありました。

「スポンサーをつけたらどうでしょう。アマチュアとして問題あるかな？」

べつに問題ないと思いますが，お金さえ出してくれれば，どこでもよいというわけにもいきません。例えば，シャープや富士通など特定の機種に密接したメーカーでは，審査が平等に行われないという誤解を招きます。CG関連のソフトを発売している会社も同様でしょう。

実は，すでにCGAコンテストにはスポンサーがついています。それは「ASAHIパソコン編集部」です。当時の編集長の矢野様が「朝日新聞という会社は，こういった文化活動も支援する会社なんだ」とおっしゃって，第3回以来，全体の予算の約3分の1程度の金額を後援金としてくださっています。また，Oh!X編集部からも，いくらか協賛金をいただいています。しかしバブル崩壊後，その情勢は厳しく，次回についてはまったく見通しがついておりません。

とりあえず，CATV各社なんかにコンテストのビデオとお手紙を送ってみたのですが，反応はほとんどありませんでした。どこか，マルチメディアを謳っている企業などが一口乗ってくださるとうれしいのですが。企業側にも，後援するメリットがないと……。なかなか難しいのが現状です。

その他のご意見も見てみましょう。

摂津市Iさん**「せっかく郵便振替になったのですから，カンパの額などを半端な数にするという考えもあっていいと思います。実費2,455円＋カンパ800円＝3,255円とか」**

これは，理屈よりも性格的な問題ですね。私は，お金に対しては大ざっぱな性格なんです。こういう発想はありませんでした。はい。

八日市市Mさん**「DōGAは営利団体ではないが，ボランティア団体でもないのであれば，ちゃんと採算がとれる，健全な運営を行うべきだ。また，私たちはその対価を払うのが当然だろう」**

ボランティア団体の意味がよくわかりませんが，DōGAってボランティア団体と違うのでしょうか。コンテストにしても，スタッフは全員バイト料なしのボランティアで運営しているんやけど。

それでは，次のアンケート。CGAシステムのマニュアルがなくなってしまったが，どうしたらよいと思うか。

A) すでに持っているので別にかまわない［7％］
B)まだ入手していないので，いまのものを増刷してほしい［19％］
C)すでに持っているが，新バージョンのマニュアルを作るなら，また申し込む［34％］
D) 大幅なバージョンアップでない限り申し込まない［36％］

まだ入手していない人や，バージョンアップを望む人が結構いるということですね。

天理市Iさん**「Oh!Xのバックナンバーを手に入れても，マニュアルがないと使いようがない。どうにかしてくれ！　マニュアルがなくなったらおしまいという態度は，もう新人はいらないという態度に等しいぞ」**

いや，新人はいらないなんてちっとも思いませんが，マニュアルの再販，再配布は手間がかかるんです。しかし，先月も通知したように，

名)にしておかなければいけません。もともとの形状をオーバーライトしないように十分気をつけてください。

準備ができたら,「parts」のなかの,その形状にとって適当と思われる分類のディレクトリに自分でモデリングしたデータをコピーします。同様にシンプル版のデータも,「sparts」のなかの同じディレクトリにコピーします。適当な分類がなければ,新たに「自作」といったディレクトリを作ってもかまいません。要は,「parts」と「sparts」のなかの構造をまったく同じにすればよいのです。

バージョンアップおよびマニュアルの最後の配布を行うことになりました。この申し込みが終わったあとで「知らなかった」とかいっても後の祭りですから,希望者はちゃんと申し込んでくださいね。詳細は別コラムを読んでください。

では,次は「どのようなデータベースが欲しいか」のアンケート。複数回答です。

1)大理石,惑星,樹木などのマッピング用画像 [64%]
2) いろいろな風景の背景用画像 [51%]
3) 家やビルなど建築物の形状データ [50%]
4) 車や飛行機などメカのデータ [48%]
5) CGA作品につけるBGMデータ [43%]
6) 家具などの模様替えシミュレーション [28%]
7) ロボットの形状データ [27%]
8) 老若男女,いろんな表情の画像 [17%]

マッピング用の画像がトップというのはちょっと意外ですね。そんなにマッピングを使いこなしている人が多いとは思っていませんでした。これらの要望は,よく検討し,今後TAKERUなどで発表していきたいと思います。

アンケートの最後は,賛助会員についてです。

A) 絶対に参加する [5%]
B) きっと参加する [14%]
C) 参加するかもしれない [27%]
D) 参加しないと思う [11%]
E) これだけではわからん [40%]

「参加するかもしれないが,これだけではよくわからん」ということですね。ごもっともです。

「賛助会員の目的がいまいちはっきりわかりません。もともと『活動に賛同し,協力の意思がある』人しかいないのではないですか。どこが違うのでしょうか。そのへんをもう少し明確にしてほしいと思います」

すいません。もともと,CGAシステムのユーザーは「賛同,協力の意思がある」ということになっていますが,誰が,どこにいて,どの程度意志があり,各自何ができるかなど,全然把握されておらず,意思はあってもその力が有効に活用される機会はほとんどありませんでした。そこで,そのあたりを整理して名簿を作るために,賛助会員という形式を取ったわけです(無断で名簿に住所や本名を載せられるのは嫌でしょ?)。できた名簿は会員全員に配布し,各地で会員同士の交流にも役立てたいと思います。

具体的な活動としては,データベースの構築にご協力いただいたり,イベントのときの現地スタッフとして活躍していただけたらと思っています。いまはバージョンアップマニュアルの制作で忙しいのですが,それが終わり次第名簿の作成作業に入ります。また,案内の手紙を送りますので,その節はよろしくお願いします。

そのほか,多かった質問をいくつか取り上げてみましょうか。

「以前のコンテストのビデオも入手できないものでしょうか?」

ぱらぱらと来る要望を,いちいち処理してたら,それだけでほかの活動はストップしてしまいます。残念ながら,大阪近辺の方で「私がビデオ発送係を専任します」という方が現れない限り無理でしょう。今回のマニュアルの再配布もそうですが,申し込む方はちゃんと期間中に申し込んでください。

「CGAシステムの商業利用を認めないというのは,どのような理由によるのか。また,商業利用とはどの程度の利用までを含むのか。総じてDōGAは権利関係の規定がルーズではないだろうか」

私としては,当チームとユーザーの関係は,企業間の契約のようにビジネスライクな関係ではなく,人と人のつきあいのように,もっと単純で,あいまいで,自然なものだと考えています。つまり,禁止をする根拠というのも,単に「こっちが無償で開発したもので,お金もうけしようというのは,そりゃセコイやないか」というだけの話です。

ですから,何が営利目的にあたるかは,形式的な問題ではなく,各自が良心と相談して決める問題です。ゲームのキャラクターをCGAシステムで作るという場合でも,あなたの目的意識によって,商業利用とも非商業利用ともいえます。

あなたの目的が,善意に基づいた慈善行為であるならば,CGAシステムを使っていただくことで我々もよい行いに参加でき,ありがたく思います。逆に,目的がお金儲けであるならば,当方だってその利益の分け前をもらってもバチは当たらないでしょう。まぁ,営利目的かどうかに関わらず,CGAシステムが価値を生み出したなら,適当にカンパいただければ幸いです。

美濃加茂市Wさん**「コンテストを年2回にしてください。3回でもいいです」**

無茶いいなさんな。とりあえず,そういうことは,年2~3本の作品作ってからいいなはれ。

相模原市Yさん**「次回は,LDがあるとうれしいです。多少高くても,LDのほうが画質はいいと思います」**

ちょっと調査したところ,もしLDを制作すると1枚1万円ぐらいになりそうです。これは「多少」の範囲に入るのでしょうか?

四条畷市Kさん**「阪大に落ちて,同志社大に行くことになりました。こんな人間でも活動に簡単に参加できるような体制を早く作ってください」**

先にも述べたように,賛助会員の組織作りを進めています。しかし,Kさんは大阪なのですから,直接当プロジェクトルームに出入りすればよいと思います。また,同志社大学のコンピュータクラブに入り,大阪大学コンピュータクラブや京大マイコンクラブのように,クラブ単位で提携する方法もあります。

名古屋市Iさん**「今回は郵便振替なので楽です。次回もこの方法でお願いします」**

そうですね。いろんな方法を行ってきましたが,これがいちばん確実なようです。でも,郵便振替用紙をつけるのは,Oh!X編集部側の都合もありますので,確約はできません。編集部のみなさん,今後ともよろしくお願いします。

東京都Sさん**「もっとコンテストを大きくやるとよい。FM TOWNSとかがなかったりするのは,X68000ユーザー以外に知名度が低いからだ」**

そのとおりだと思います。しかし,問題は,どうやったら知名度が上がるかです。何かよいお知恵はないでしょうか。

東京都Uさん**「かまたさんの文章のなかに,ときどき大阪弁が混じりますが,いい味出していると思います。さすがに大阪の人ですね」**

ごめんなさい。私はいままで読者の皆さんをだましてました。実は,私は生粋の大阪人ではありません。実は東京生まれなんです。東京にいたのは2~3歳ぐらいまでですが,小学校まで関東なまりが抜けませんでした。子どもの頃,「おまえ,何いきってるねん!」といわれたりしましたが,「いきってる」の意味がわからず戸惑ったこともあります(いきってる:カッコをつけてる)。

仙台市Sさん**「作成した画像を実際にビデオテープに録画するときの手順を詳しく特集してほしいです」**

そうですね。このへんのノウハウは,結構奥が深く,より簡単に,高画質にするにはテクニックが必要です。でも,この連載のネタはだいたい半年先までつまっているので,いますぐには難しいと思います。皆さんもほかにやってほしいテーマがありましたら,お手紙ください。

東京都Mさん**「CGAシステムの最新バージョンは,TAKERUでいつでも手に入るようにして,マニュアルもタケルの説明書の紙でプリントアウトする方法はいかがでしょうか」**

むちゃいいまんなぁ。800ページのマニュアルをその場でプリントアウトできますかいな。

岐阜県Eさん**「CGAマガジンなどを集めてビデオで出してほしい。X68000では作画する気にならないよ。CGAするうえで作画がいちばんのネックです。シャープよ,爆裂68を出してくれ!!」**

そーだ,そーだ。ところで,DōGA内では,近く計測技研のLANユニットを導入して,486マシンとX68000をつなぎ,作画はできるだけ486マシンにさせるシステムを構築するつもりです。うまくいけば,試用レポートなど紹介するかもしれません。

* * *

ということで,各種お問い合わせに答えさせていただきましたが,これでも紹介できたのはほんの一部です。でも,お手紙の類は全部ちゃんと目を通しています。ただ,こちらも忙しいので,なかなか全部に返事を書けません。ですから,お手紙には必ず電話番号と,いらっしゃる曜日,時間帯などが明記されていると助かります。また,電話を家族と共有されている場合,返事の電話があるかもしれないといっておいてください。ときどき,変な団体と勘違いされるんだもん(笑)。これからもよろしく。

よく使う組み合わせの登録

「基本(プリミティブ)」や「細部(ディテール)」をたくさん組み合わせて，たとえばエンジンや艦橋を作ったとしましょう。そのエンジンや艦橋を，ほかのメカで使用しようとすると，もう一度組み直さないといけません。しかし，上記の「いらないデータの削除」と「オリジナルパーツを加える」の知識を生かせば，この問題は解決します。

新たに作ったエンジンを，たとえば「eg12」という名前にしておきます。この形状は「SUF」のなかの「eg12.SUF」ですから，これを「parts」のなかの「エンジン」にコピーします。同様に，「SSUF」のなかの「eg12.SUF」がシンプル版なので，これを「sparts」のなかの「エンジン」にコピーします。そして，最後に「SUF」や「SSUF」のなかの不要なデータを削除します。

複数のカットのアニメーション

「GENIE」では，1カットずつしかアニメーションすることができません。まぁ，それでもビデオにつなぎ録り

発表！「GENIE」コンテスト

飯：はい，ハイスクール飯干です。
中：はい，アップル中野です。飯干君と漫才のコンビを組んでます。
飯：誰が漫才のコンビやねん。しかし，ホンマになんでんなぁ，7月号で募集した「GENIEでこんなの作っちゃったコンテスト」にたくさんの応募をいただきまして，ほんとにありがたいこっちゃなぁ。
中：こんな名前のコンテストやったっけ？
飯：細かいことはええの。では，面白い作品，気に入った作品を紹介しましょう。
中：たくさんあるから迷うなぁ。まずは，ごく普通に，宇宙戦艦と宇宙戦闘機からいくつか選んでみようか。
飯：そしたら，八尾市の藤沢和行さんのデザインがかっこいいと思うな。「CRUSHER」に「BASE」。
中：「CRUSHER」は，映画「クラッシャージョウ」に出てきたメカをイメージしたそうです。
飯：なんや，オリジナルメカとちゃうん？
中：いや，「なんか，こんな感じやったなぁ」という程度にしか参考にしてないそうやから，オリジナルといっていいと思うよ。
飯：このメカを作るためのデータは，リスト1にまとめています。ただし，左右対称のメカは，右半分を省略しました。物体名のあとに「@」がついているものは，右側(Y座標が－)にも作ってください。
中：このデータの特徴としては，「CR08」をスラッと細目にしてるところなんかバランスがいいと思うねん。
飯：リストを見ると，エンジンまわりも面白い使い方をしてるで。「WG12」に元からついているエンジンに，太めの「PL01」を重ねたり，「EG08」をつぶしてウイングにしたり。
中：ほんまやなぁ。写真ではわかりにくいけど，「BR01」を上下逆にしてジョイント代わりにしているとか。人のデータを見ると勉強になるなぁ。
飯：「BASE」のほうは，本人は特徴がないといってはるけど，「BS04」と「BS06」を直角に交わらせるなんて大胆やなぁ。
中：「BR06」を船体に食い込ませているのも面白い使い方やね。えっと，ほかには，名古屋市の小田島さんの「SF_001」(リスト2)なんかもカッコいい。
飯：特に凝った工夫をしてるわけじゃないけどね。
中：ほかには，栃木県の平山さんの作品なんかも好きやな。
飯：えっ平山さん？ CGAマガジンでエンジンを作ってくれた人ではないですか。
中：あの，ノーマル直列4気筒エンジンと，ターボ直列4気筒そしてロータリーエンジンとか……？ それやったら，「GENIE」使わんでも，メカ作れるやん。
飯：平山さんも，まさかこんなに簡単にできるとは思わなかったとのことですが，そりゃロータリーエンジン作れるんやから，当然やわな。
中：たくさん送っていただいていますが，そのなかから「S_CRU」と「SS_P」を紹介しましょう。
飯：「S_CRU」は宇宙巡洋艦，「SS_P」は民間の惑星間連絡船ということですが，そういう感じには，ちょっと見えないなぁ。
中：でもこの「SS_P」は，なんかNASAっぽくていいぞ。
飯：なるほどねぇ。「GENIE」のデータにも，NASAっぽいのがもうちょっとあってもよかってんけどね。
中：このエンジン部分，「BS11」を3個使ってるで。「BS11」は1つでも497面あるのに。よう，メモリ足りたなぁ。
飯：そりゃ，ロータリーエンジン作るような人やもん。メモリはたくさん持ってるわ。
中：じゃぁ，「SS_P」はおいといて，「S_CRU」のデータをリスト3に掲載しましょう。
飯：ほかに宇宙戦艦で面白そうなのないかなぁ。
中：面白そうという意味では，これなんか，かなりいい線いっているぞ。
飯：どなたの作品？
中：それが，ラベルにもドキュメントにも書かれてなかったので，わからんねん。
飯：本当？ ちょっとドキュメント見せて。
「えー，初めてCGAなるものに触れたわけですが，いや，これは楽しい！ 私は，巨大なものが好きで，パーツを拡大するだけでも喜びに浸れてしまう。そんなわけで，相当デカイ戦艦ができました。ところで，CGというのは本当にメモリ食うのですね。」
飯：こんだけくっつけたら，メモリ食うわな。ということで，「PEM_G」です。でも，やたらくっつけているけど，デザインとしては洗練されてないような気もするけど。
中：そうか？ 結構好きやで。特にアップにすると，迫力あるもん。
飯：当然ですけど，このメカのリストは掲載できません。
中：そのほか，かっこいい戦闘機や戦艦というと，アレですか。

リスト1 CRUSHER

```
{ mov (  400 200  -50 ) scal( 1.25 0.4 0.5 )   obj cr09 @ }
{ mov ( -700   0    0 )                        obj wg12 }
{ mov ( -500   0    0 )                        obj cp04 }
{ mov ( -700 500 -100 ) scal( 0.5  0.8 0.8 )   obj pl01 @ }
{ mov ( -550 500   50 ) scal( 0.8  0.5  -1 )   obj eg08 @ }
{ mov ( -550 650 -100 ) rotx( 90 ) scal( 0.8 0.5 1.0 ) obj eg08 @ }
{ mov ( -550   0    0 ) scal( -0.75 0.75 -1 )  obj br01 }
{ mov ( -300   0 -400 ) scal( 0.5 0.5 0.5 )    obj cr08 }
{ mov ( -950 500 -100 ) scal( 0.75 0.24 0.24 ) obj eg06 @ }
{ mov (  500 250  -25 ) scal( 1.5 1.5 1.0 )    obj dt03 @ }
{ mov (  500   0  -25 ) scal( 0.75 4,0 0.5 )   obj dt05 }
```

リスト2 SF_001

```
{ mov (    0    0    0 ) obj cp04 }
{ mov (    0    0   12 ) obj eg05 }
{ mov (    0  200    0 ) obj eg09 @ }
{ mov (    0  250    0 ) obj sl06 @ }
{ mov (  180    0 -130 ) scal( 0.75 0.4 0.3 ) obj cn06 }
```

リスト3 S_CRU

```
{ mov (    0  300    0 ) obj pl03 @ }
{ mov ( -200    0    0 ) rotx( 90 ) scal( 0.8 1.0 1.5 ) obj pl09 }
{ mov ( -400    0   50 ) rotz( 90 ) scal( 0.2 1.0 0.2 ) obj ot10 }
{ mov (    0    0  -50 ) scal( 1.5 0.5 0.5 ) obj pl04 }
{ mov ( -700  300   50 ) obj eg08 @ }
{ mov ( -800    0    0  ) scal( 1.0 0.5 2.0 ) obj jt08 }
```

していけば，どんなに長いアニメーションでもできますが，ちょっと面倒です。そこで，「GENIE」を終了し，直接HANIM.Xを使って，複数のカットをアニメーションしてやりましょう。

アニメーションを実行するためには，どの画像をどういう順番で表示するかを定めるタイムチャートファイル（＊.TCH)が必要です。このファイルは簡単な書式ですので，エディタで書いてもよいのですが，MKTCH.Xを使って自動的に作ってみましょう。

「FSC」のディレクトリに入り，「SAMP1」「SAMP2」「SAMP3」がすでに作画されているとします。そして，この順番に連続してアニメーションさせる場合，

MKTCH SAMP1¥SAMP1 SAMP2¥SAMP2 SAMP3¥SAMP3

とします。すると，「SAMP1.TCH」というタイムチャートファイルができます。そして，

HANIM /M2 SAMP1

を実行すると，しばらく画像を読み込んだのち，アニメーションが始まります。ただし，メモリが十分にないと，すべてを読み込むことができません。上記の例ですと，画質をどのように設定させて作画したかにもよりますが，フリーエリアがだいたい1Mバイトほど必要になります。

飯：アレやねぇ……。でも，ちょっと，かまたさんに相談しなきゃ。
か：だめ！　著作権にひっかかるのは，全部ダメ。誌上では公開できません。
中：え〜，富永さんのNENESISとか，藤沢さんのシルフィードとかも……？
か：当然だめ。
飯：尾形さんのブロッケード・ランナー(STAR WARS)なんかメチャメチャ凄いでぇ。なんと，パーツ数368個，ポリゴン数7647面！
か：エライ力作やなぁ。ちょっと見せて。オー，こいつは凄い。でも，ダメ。
飯：ねぇ，絶対だめ？　ふん，すねちゃうぞ。
中：なんでもええけど，そろそろ帰らな，電車なくなるで。
飯：おっ，もうこんな時間か。じゃ，そゆことでかまたさん，あとはよろしく。
か：なっ，なんつう，無責任な奴ら。仕方がないので，引き続き紹介させていただきます。宇宙船以外のメカもいろいろありました。まずは，藤沢さんの「BIKE」。前後のサスの使い方がカッコイイですね。「TK05」をY軸方向に0.1倍して薄くしているんですか。エンジンもそれっぽいし。なんといっても全体のバランスがいい。でも，ハンドルがないですね。まぁ，いいけど。このデータはリスト4に掲載します。

リスト4を見ると，回転も多用してますし，拡大の値も微妙な数値になっています。試行錯誤されたんでしょう。

続いて，また藤沢さんの労作「TANK」。このタンク，車体部はなかなかよいですが，砲台部が「CN05」そのものじゃないですか。それに，ちょっと上が重すぎるような気も……。それはともかく，注目してほしいのがキャタピラ。どうやって作ったかって，ただひたすらプリミティブを並べたそうだ。そりゃ，大変だ。パーツ数が118もあるので，これもリスト掲載は省略します。

ほかに変わったものといえば，メカ動物(？)があります。まずは，埼玉県の小野寺さんの「INU6」。犬の顔だそうです。確かに，そう見えないこともないけど(メスライオンという気もする)，「GENIE」でこんなものを作るとは思いませんでした。

続いてもう1つ，松本市の行本さんの「OST」。駝鳥のようですが，一応，多脚型強襲揚陸艦「まりりん」なんだそうです。お手紙によると，自分で作ったメカには名前を全部つけて，勝手に世界観を想像して楽しんでいるとか。

さて，最後に意外と多かったロボットものからいくつか紹介しましょう。倉敷市の岡田さんの「REPORT」は，ガンダム系のデザインでしょうか。胴体や顔はちょっと変わってますが，手足がうまくできていると思います。

取手市の吉田さんは，エレクトリックシープの「ロボットコンストラクションR.C.」というゲームで自分がデザインしたロボット「KISBEY」を作ってみたそうです。4輪にレーザーとマシンガンを装備しているとか。

岡崎市の山本さんの「HAN」は，ロボットというより，モビルアーマーという感じですが，コメントがないのでよくわかりません。デザインもよくわからない形をしているのですが，なんとなくカッコイイ。センスもオリジナリティもあると思います。

そして，ラストはもう一度藤沢さんの作品，「ROBO」です。これもかなり変わったデザインですが，やっぱりセンスは抜群です。ちなみに「ROBOF」は飛行形態なんだそうですが，あまりよくわかりません。この「ROBO」のデータは，リスト5で紹介します。

＊　　＊　　＊

さて，いろいろな作品を紹介してきましたが，審査の結果は，「CRUSHER」「BASE」「BIKE」「TANK」「ROBO」を制作された八尾市の藤沢さん，質・量ともに群を抜いていて，グランプリに決定です。おめでとうございます。賞品としては，「大阪名物・阪神タイガース応援用ハッピ」をプレゼントしようと思っていたのですが，藤沢さんは大阪の人です。大阪人なら，一家に数着，すでにあるに決まってますので，ほかのものを検討いたします。期待せずに，お待ちください。

なお，今回応募したのに紹介されなかった方，ごめんなさい。さぁ，次のコンテストは，第7回CGAコンテストだ！　締め切りは，毎度お馴染み12月31日。いつもは多少遅れても受けつけていたけど，今年は締め切り厳守だ。気をつけろ。それから，9月末ぐらいに，とりあえずできているところだけ(極端な話，脚本だけでもいいぞ)送ってください。仕上げに向けて，当方からアドバイスをするというサービスを行います。希望者は「コンテストアドバイス希望」係まで。

リスト4　BIKE

```
{ mov (  700    0 -100 ) roty( -130 ) scal(  0.5  0.3  0.5 ) obj pl08 }
{ mov (  900    0 -350 )              scal(  0.4  0.2  0.4 ) obj ot05 }
{ mov ( -100    0  100 )              scal( -0.5  1.2  1.0 ) obj cr09 }
{ mov ( -700    0  100 ) roty( 20 )   scal( -0.5  0.75 0.5 ) obj cr05 }
{ mov (  200    0 -150 )              scal(  0.5 0.75 0.75 ) obj ot03 }
{ mov ( -100  100 -150 ) rotz( 90 ) rotx( 90 ) scal( 0.25 0.25 0.25 ) o
bj ot09@ }
{ mov ( -100    0 -150 )              scal(  0.4  0.4  0.4 ) obj bs12 }
{ mov (  250    0  150 )              scal(  1.0  0.8  2.5 ) obj cp03 }
{ mov ( -900    0 -300 )              scal(  0.5  0.2  0.5 ) obj ot05 }
{ mov ( -600  100 -100 ) roty( -30 )  scal( 0.75  0.1  1.0 ) obj tk05@
 }
{ mov ( -300    0 -300 ) rotz( 90 )   scal( 0.65  0.5  0.5 ) obj cn02 }
{ mov ( -300    0 -300 ) rotz( -90 )  scal( 0.65  0.5  0.5 ) obj cn02 }
{ mov ( -100 -150 -350 )              scal( 1.5  0.75 0.75 ) obj tk01 }
{ mov ( -100 -100 -300 ) rotx( 90 )                          obj dt06 }
```

リスト5　ROBO

```
{ mov (    0    0  600 )              scal(  0.6 0.75  0.5 ) obj cr05 }
{ mov ( -200    0  650 )              scal( 0.75  1.0  1.5 ) obj eg08 }
{ mov (  -50    0  300 ) roty(  45 ) scal(  0.3  0.5  0.5 ) obj pl02 }
{ mov (  -50  200  150 ) rotz( -20 ) roty( 110 ) scal( 0.5 0.25 0.3 )
obj tk04@ }
{ mov ( -500  250 -100 ) roty(  10 ) obj eg09@ }
{ mov ( -600  250 -250 ) roty( -80 ) rotx( 90 ) scal( 0.2 0.6 0.3 ) ob
j eg04@ }
{ mov (  200  350 -500 ) rotz( 20 ) roty( -130 ) scal( 0.5 0.5 0.5 ) o
bj pl08@ }
{ mov (  400  400 -900 ) roty( 87 ) rotx( -110 ) scal( 0.5 1.25 1.0 )
obj wg07@ }
{ mov (    0    0  250 )              scal(  0.3  0.5  0.3 ) obj jt12 }
{ mov (  -50  400  700 ) rotx( 20 )  scal(  0.3  0.5  0.3 ) obj tk02@
}
{ mov (  -50  550  200 ) rotz( 90 ) roty( 90 ) scal( 0.65 0.75 0.75 )
obj tk01@ }
{ mov (  -50  550  800 ) rotz( 90 ) roty( -20 ) scal( 1.0 0.5 0.5 ) ob
j cn02@ }
{ mov (  -50  550  350 ) roty( 90 )  scal(  0.3  0.3  0.3 ) obj sl06@
}
{ mov (  100    0  500 ) roty( 80 )  scal(  0.5  0.5  3.0 ) obj dt02 }
{ mov (    0    0  800 )              scal(  0.4  0.5  0.5 ) obj ft05 }
```

より高度な使い方

「GENIE」を使えば，誰にでも宇宙戦闘シーンのなかの1カットができるでしょう。しかし，それはあくまでも1カットであり，戦闘シーンではありません。「えっ，いま複数のカットをアニメーションさせる方法を解説したところじゃないか」という方もいると思いますが，複数のカットを連続してアニメーションしたからといって，それがちゃんとしたシーンになるとは限らないのです。「GENIE」の最も高度な使い方とは，「映像」を作ることにほかなりません。試しに，作った数カットを連続してアニメーションしてみてください。下手をすれば，それはばらばらの数カットにしか見えず，それぞれ別々に見たときと変わりません。しかし，うまくつなげれば，1枚ずつ見たときの数倍も迫力を増すことができます。

さらに，なんの意味もなかったカットを組み合わせることで，たとえば出撃前の緊張感とか，勝利の凱旋といった別の意味を持たせることができます。このような手法を映像の専門用語でモンタージュといいます。

そこでぜひひとつチャレンジしてほしいことがあります。簡単なストーリーを用意し，そのシーンを自分で作ってみることです。

ストーリーは，たとえば「不利な戦況下，新型兵器を積んだ主人公の戦闘機が空母から発進し，敵の防衛線を突破，敵母艦に突っ込んで自爆する。その活躍で救われた味方の艦隊は主人公に感謝しつつ帰還する……END」ちょっと長すぎますね。これの一部でもいいでしょう。

このシーンに台詞は不要です。すべて映像だけで説明できるはずです。どのようなカットを作れば，見ている人にストーリーが理解されるか。また，思わず感情移入するような緊迫感を持たせることができるか。

どのようなカットをどのような順番で見せるべきか，よくよく考えて，そして実際に作ってみてください。そして，できたカットをつなげてアニメーションさせて，第3者に見てもらいましょう。ちゃんと理解してもらい，最初から最後まで緊迫感を維持できたとすれば，あなたは間違いなく天才です。プロを目指そうが，CGAコンテストのグランプリを狙おうがご自由にしてください。

しかし，たいていの場合は，わけのわからないカットの寄せ集めになるのが関の山です。友達に見せても，「なんやようわからん」といわれ，途中で飽きられてしまうでしょう。別に，あなたに才能がないわけでなく，初めて映像を作ったら，誰でもその程度です。

しかし，皆さんに知ってもらいたいのは「映像」の奥の深さ，楽しさです。ぜひともそこで諦めずに，どこが悪いのか考えて，何度も試行錯誤してみてください。

具体的にどうすればよくなるかは，ページ数の都合によりここでは省略します。とりあえず，CGAシステムのマニュアルのT-193ページ「映像理論概論」，T-291ページ「映像理論研究」を読んでみてください。きっと，参考になると思います。

おわりに

「GENIE」の解説は今回で終わりです。来月は，1回お休みさせていただきます。なんといっても，マニュアルの追加版を作成しないといけませんから。11月号では，新しくマニュアルを手に入れた初心者向けの企画を考えています。また，お楽しみに。

マニュアル申し込み締め切り迫る！

先月号でもお伝えしたように，CGAシステムのマニュアルの再配布，およびバージョンアップサービスを行うことになりました。今回が最後の配布となりますので，希望者はこれを逃さないように，繰り返しご注意申し上げます。締め切りまで，あと数日だ！　急げ！

［申し込み内容］
・マニュアルコース
　旧マニュアル(800ページ)
　追加マニュアル(約100ページ)
　最新プログラム
・バージョンアップコース
　追加マニュアル(約100ページ)
　最新プログラム

［実費］
マニュアルコース　　　：5,000円
バージョンアップコース：2,500円

［申し込み期間］
1994年7月18日～8月31日

［発送期間］
9月中に印刷，発送は10月以降

［申し込み方法］
Oh!X 8月号綴じ込みの郵便振替用紙，または通常の郵便振替用紙を使用。
・以下は，通常の郵便振替用紙での申し込み方です。8月号の綴じ込み用紙を使用する方は，8月号の注意事項を読んでください。

［記入事項］
・当方の口座番号：大阪3-109598
・加入者名：DOGA（当方の住所等は必要なし）
・金額：振り込む金額を記入してください。実費に加え，カンパも同時に受けつけます。
・「払込人住所・氏名」欄には，ご自分(発送先の)住所，氏名，電話番号を丁寧に明記してください。難しい漢字にはふりがなをふってください。
・第6回のビデオを申し込まれた方は，ビデオの封筒のラベルに書いてあったDōGA登録ナンバーを，表面の「払込人住所・氏名」欄の下のほうに記入してください。
・裏面の通信欄に「マニュアルコース希望」「バージョンアップコース希望」のどちらかを記入してください。
・3.5インチディスクを希望される方は，通信欄に「3.5インチ希望」と明記してください。記入がなければ，5インチとなります。
・そのほか，ご意見，ご感想などをご自由にお書きください。

［注意事項］
・コンテストのビデオなどを同時に申し込むことはできません。マニュアル及びバージョンアップだけです。
・郵便振替用紙1枚につき1件の申し込みとします。複数申し込まれる場合は，複数回に分けて申し込んでください。
・CGAシステムは，当チームの活動の主旨に賛同し，参加，または協力の意志がある方にのみ使用が許され，営利目的には使用できません。あらかじめご了承ください。
・上記の費用は，マニュアル印刷，発送の実費であり，いわゆるプログラムの値段ではありません。プログラムの値段は，基本的に無料ですが，カンパは随時受けつけています。
・上記事項を守られていない方の入金は，ただの全額カンパとして処理されます。

WE WANT YOU!

Oh!Xは，読者の皆さん1人ひとりの力が作り上げていく雑誌です。あなたも誌面作りに協力してくれませんか？

投稿大募集

Oh!Xでは読者の皆さんによる投稿作品を常時募集しています。

未発表の作品であれば，グラフィック，音楽，システムプログラム，ツール，ゲーム，ハードウェアなどジャンルを問いません。機種についても特に限定はしませんが，雑誌の性格上扱いにくい場合もあります。

誌面に載りきらない大きなアプリケーションなどはディスクメディアを使って配布することが考えられます。その形態のひとつはご存じ付録ディスク，そしてもうひとつは別冊形式によるものです(発売中の「Z-MUSICシステムver.2.0」に続き，今後もいくつかのOh!X BOOKSシリーズが予定されています)。

また，「こんなものを作ってみました」といったものでもかまいません。気軽に作品を送ってみませんか。

投稿募集要項

1)　お送りいただくプログラムには，住所，氏名，年齢，職業，連絡先電話番号，機種名，使用言語，動作に必要な周辺機器，パソコン歴などを明記のうえ，封書の宛先の最後には「Oh!X LIVE」「全機種共通システム」「投稿ゲームプログラム」など，プログラムの内容を明確にご記入ください。

2)　投稿されるプログラムには詳しい内容を記入した原稿を同封してください。ディスクの中にドキュメントファイルの形式でのみ記述している方がいますが，郵送時の事故などでメディアが破壊されることもありますので，必ず文書を添えるようにしてください。変数表，メモリマップ，参考文献などの情報があればなお結構です。また，掲載に際しては，プログラムやデータ原稿に対して加筆修正をさせていただくことがあります。

3)　お送りいただくプログラムは事故防止のため最低2回はセーブしておいてください。基本的に原稿などの返送はいたしませんので，あらかじめご了承ください。

4)　ハード製作関係の投稿については，最初は内容のわかる原稿のみお送りいただければ結構です。その後，当方で製作物が必要だと判断した場合には改めてご連絡いたします。

5)　作品の採用については，掲載号が決定した時点で当方より連絡いたします。特にツールやハード関係などの作品は特集内容などを考慮したうえで採用決定されますので，結果を連絡するまで時間がかかる場合があります。

6)　投稿いただいたプログラムにバグなどが発見された場合は，新しいプログラムの入ったメディアと一緒に文書にてご連絡ください。

7)　掲載されたプログラムに対しては当社規定の原稿料をお支払いいたします。また，投稿されたプログラムの著作権などはすべて制作者に保留されますが，いわゆる「フリーソフトなどとしてネットにアップする」ことなどを希望される場合には，必ず事前に編集部までご連絡ください。なお，一般的モラルとして，他誌との二重投稿，または他誌に掲載されたプログラムの移植などは固くお断りいたします。

その他，不明な点は編集部までお問い合わせください。

Oh!X編集部　☎03(5642)8122

協力スタッフ募集

Oh!Xでは誌面作りに参加していただく協力スタッフを募集しています。

スタッフとして活動する熱意があり，東京近郊にお住まいの方でソフトバンクに来社可能な方。時間的束縛は特にありませんが，ある程度時間に余裕がある方に限ります。基本的に学生を対象にしていますが，時間的余裕と余力が十分にあれば社会人も可とします。ただし，18歳未満の学生および浪人生の方については採用予定はありません。

応募要項ですが，ライター希望の方はOh!X誌面1ページ分相当(2500字程度)の自由論文に自己紹介文を添えて「Oh!Xスタッフ希望」係までお送りください。

また，文章力には自信がないけどプログラムなら……という方でも技術スタッフとして参加していただく場合があります。こちらを希望の方は，自由論文の代わりにこれまでに制作した自作プログラムとその解説などを一緒に応募してください。

書類選考後，採用者の方にはこちらからご連絡いたします。

すべての読者へのお願い

いまはまだ何もできないけれど，いつかは……と思っているアナタにも，いますぐできるいちばん重要なことがあります。アンケートハガキへのご協力です。Oh!Xの誌面の方向性は，このアンケートで寄せられた読者のご意見をもとに決定されています。

皆さんからの熱いメッセージをお待ちしています。

そして，宛先

〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク株式会社
Oh!X編集部　○○○○係

イラスト投稿の規定

サイズはハガキ大(A6判)からB5判くらいまでを目安としますが，取り扱いの手間や現実的な問題としてハガキ大を一応の標準とします。いずれにせよ，掲載時にはかなり縮小されることを考慮して描いてください。

一応の推奨形式は以下のとおりです。

1)　ハガキ大のケント紙で郵送

ハガキでも結構ですが，たまに裏面にも消し印が押される危険があります。

2)　黒一色(薄ズミ不可)

墨汁は汚れの原因になることがあります。製図用インクがおすすめです。原稿は縮小されますのでスクリーントーンの80，90番台(レトラセットの場合)や色の濃すぎるものなどについての再現は保証しかねます。また，残念ながら，カラー原稿はごくたまにしか掲載されません。

内容に関して特に規制はありませんが，季節ものについては，掲載が予想される時期を考慮して早めに送ったほうが有利になることがあります(年賀状は例外)。

皆さんの力作をお待ちしております。

Oh!X LIVE in '94

X68000・Z-MUSIC ver.2.0用
(SC-55mkII対応)

LOVE IS ALL

I'VE NEVER BEEN TO ME

Uchiyama Toshihiko
内山 利彦

X68000・Z-MUSIC
ver.2.0用

「HELL HOUND」より 季節風

Komatsu Yasurou
小松 恭郎

X68000・Z-MUSIC用

踏切の通過音

Hasunuma Masaru
蓮沼 勝

日本では秋が結婚シーズンだそうなので，季節ものってことになるのでしょうか。椎名恵のあの曲です。ゲームファンには先月号に掲載の「HELL HOUND」。これを聴くとゲームもしたくなる!? あとは，なかなか好評のイロモノもあります。

愛こそすべて

さて，今月は美しいバラードで泣いていただきます。椎名恵のアルバム「ballads」より「LOVE IS ALL」です。いまじゃブライダルシーンを飾る定番となったこの曲，歌詞もメロディも，幸せの門出に花を添えるのにピッタリ。家族や友人の結婚式で聴いた方もいるはずです。え？ 自分たちの式で使った？ いや～んもう勝手にして！

演奏には，Z-MUSICシステム(ver.2.0以降)とSC-55系のMIDI楽器が必要です。制作にはSC-55mkIIが使われていますが，SC-55でも問題なく演奏できるでしょう。

内山君はZ-MUSIC初体験だそうですが，過去にMMLの経験はあるのでしょうか。ソツのない演奏で，初めてとは思えない仕上がりです。あのピュアな歌声にクラリネットを当てたのは正解でしたね。少し音が足りない気もしますが，SC-55であんまり楽器を登場させても音が足りなくなりますから，これくらいでいいのかもしれません。

ballads

この作品は，内山君がお姉さんの結婚式で実際に使ったんだとか。「楽譜がなかったから耳コピです」なんて，思わず姉弟の深い愛を感じずにはいられませんね～。

ゲームミュージックだ！

お次はオリジナルのゲームミュージック。先月のTHE USER'S WORKSで紹介された「HELL HOUND」より「季節風」です。別々に投稿されたものですが，両方採用とはなかなかニクイですよね。

演奏にはZ-MUSICシステムが必要です。内蔵音源ですがPCM8.Xはいりません。

曲はそれっぽい作りで，フュージョンのエッセンスをふりかけた純ゲームミュージック風。狙ったのかどうかわかりませんが，大事なポイントはうまく押さえてありますね。思わず口ずさんでしまうフレーズがお気に入りです。ドキュメントには「北欧の針葉樹林帯を疾走するイメージで作った」とありますが，暗黒の宇宙とカラフルな星，そして逆火山もよく似合う気がしました。

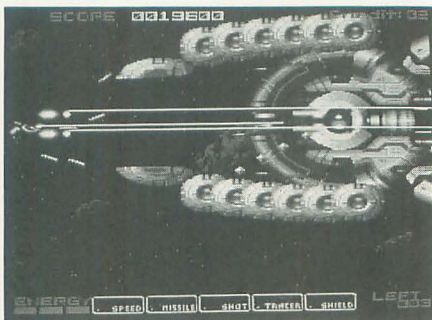
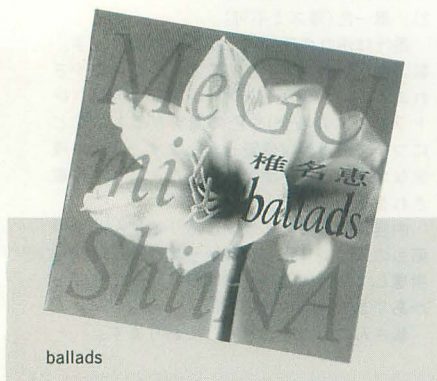
HELL HOUND

私も「HELL HOUND」プレイしましたが，あの当たり判定にはちょっとビックリ。その点をクリアすればサクサク進めてテンポのいいゲームだと思います。もう少し歯応えがあればマニアも喜ぶ……かな？

バーチャルリアリティ？

最後は，LIVE in始まって以来の「バーチャル踏切サウンド」をお届けします。そこの君，どこがバーチャルなんだ～とか怒らないように。いや～，まさかこんなネタが投稿されるとは予想だにしていませんでした。なんといっても，あの「カンカンカン」ですからね。いやはや……。

この手のネタで重要なのはリアルさなんですが(それしかないという話……)，この作品は踏切サウンドにステレオドップラー効果を導入し，立体感溢れる疑似3D空間へ貴方を誘ってくれます。物理現象をシミュレートするあたりポイント高いね。ステレオリバーブをほんの少しかけて聴くとさらにリアル。こりゃ一本取られたって感じかなぁ。ZPLKを活用してフェードアウトさせたり，すれ違う電車の音を入れてみるのも面白いかもしれませんね(PCMがステレオじゃないと無理ですね……)。

演奏は古いZ-MUSICでもOKですが，効果音のPCMデータと，バッチファイルで使

用するZPLK.Rは，どちらも「Z-MUSICシステムver.2.0」に入っています。必ずPCM入れて聴いてね。

このコーナーではこういった効果音ネタも歓迎します。ただし，いい加減に作ったようなものは採用できません。根底に流れるもの，それは究極のリアリティを追い求める飽くなき探究心であるべきなのです(あ，歯が浮いた)。　(T.F)

リスト1　LOVE IS ALL

```
  1: .comment                        LOVE IS ALL [SC-55]
  2:
  3: (i)                            /
  4: (b1)                           / L O V E   I S   A L L   [椎名 恵
  5: (o126)                         /  - I've never been to me -
  6: (m1,5000)(amidi1,1)            / 日本語詞  : 麻生圭子
  7: (m2,3000)(amidi2,2)            / 作詞・作曲 : Ken Hirsch ・ Ron Miller
  8: (m3,3000)(amidi3,3)            / 編曲      : 戸塚 修
  9: (m4,3000)(amidi4,4)            / from the album "ballads"
 10: (m5,3000)(amidi5,5)            /
 11: (m6,3000)(amidi6,6)            / copy by 内山 利彦
 12: (m7,3000)(amidi7,7)            / Module SC-55mkII
 13: (m8,3000)(amidi8,8)            /
 14: (m9,3000)(amidi9,9)
 15: (m10,3000)(amidi10,10)
 16: (m11,3000)(amidi11,11)
 17: (m12,3000)(amidi12,12)
 18:
 19:
 20:
 21: /---- 初期化 ------------------------------------------
 22:
 23: .roland_exclusive $10,$42={$40,$00,$7f,$00}
 24:
 25: / PRAT9 DRUM
 26: .roland_exclusive $10,$42={$40,$19,$15,$02}
 27:
 28: / REVERB SET
 29: .sc55_reverb $10={$03,$03,$00,$4b,$48,$00,$00}        /75
 30: / CHORUS SET
 31: .sc55_chorus $10={$02,$00,$2d,$18,$50,$03,$13,$00} /45
 32:
 33: / RESERVE
 34: .sc55_v_reserve $10={3,3,1,1,1,4,2,0,1,3,2,3,0,0,0,0}
 35:
 36: (t1)       n1 @i$41,$10,$42
 37: (t2)       n2 @i$41,$10,$42
 38: (t3)       n3 @i$41,$10,$42
 39: (t4)       n4 @i$41,$10,$42
 40: (t5)       n5 @i$41,$10,$42
 41: (t6)       n6 @i$41,$10,$42 @y1,32,67 @p108
 42: (t7)       n7 @i$41,$10,$42 @y1,32,68 @y1,102,89 @p108
 43: (t9)       n9 @i$41,$10,$42
 44: (t10)      n10@i$41,$10,$42
 45: (t11)      n11@i$41,$10,$42 @y1,102,74 @p49
 46: (t12)      n12@i$41,$10,$42 @y1,102,74 @p49
 47:
 48: (t1)       @e50 , 0 @v123 @u108 [k.sign +c,+d,+f,+g,+a] r4
 49: (t2)       @e60 , 0 @v125 @u115 [k.sign +c,+d,+f,+g,+a] r4
 50: (t3)       @e45 , 0 @v124 @u90  [k.sign +c,+d,+f,+g,+a] r4
 51: (t4)       @e20 , 0 @v95  @u88  [k.sign +c,+d,+f,+g,+a] r4
 52: (t5)       @e10 , 5 @v123 @u113 [k.sign +c,+d,+f,+g,+a] r4
 53: (t6)       @e75 ,65 @v107 @u90  [k.sign +c,+d,+f,+g,+a] r4
 54: (t7)       @e70 ,65 @v85  @u80  [k.sign +c,+d,+f,+g,+a] r4
 55: (t9)       @e105,60 @v105 @u105 [k.sign +d,+f,+a]       r4
 56: (t10)      @e90 ,25 @v118 @u115                         r4
 57: (t11)      @e68 ,20 @v120 @u100 [k.sign +c,+d,+f,+g,+a] r4
 58: (t12)      @e55 ,20 @v116 @u90  [k.sign +c,+d,+f,+g,+a] r4
 59:
 60:
 61: /------------------------P I A N O
 62: (t1)       o4 l8 q7 i0@2
 63:    gag<f4> u-30 gag gag<f4>gau+15gu
 64:    gag<f4>ga u-15 g u-15 gbg<e4>gb u-20<e>
 65:    u+50 'fb<d',1 u-10 fb<d> df<c4>
 66:    'dgb' dg 'd4fg' u-15 d u-15 g4 u
 67:    'g<ce' g<c> u-15 'g4<ce' u-15 g<ce> 'a<cf'
 68:    f<c>a4 fa u-20gu 'dgb' dg 'd4gb' u-15 db<c>u 'gb<d'
 69:    u-25 db 'f4a<c' d8 u+5 'f4b<d' u
 70:    'gb<e' u-20 gb< u+10 e> u-10 egb<e> u 'a<cf'
 71:     u-25 fa<cf2>u 'fb<d' u-20 fb u 'fb<d' u-10 dfb<c>u
 72:    'gb<d' gb 'f4gb' fd>b
 73:    'f!4.g<c'<'f!4g'>g8< f!fg4.f!4.
 74:    u-15 ga'egb'>b<e'e4.gb' u-15 ga
 75:    'e2gb' u-15 'f2gb<c' u 'fb<d'f<d> 'b4<df'
 76:    u-15 f'b<d4'u 'fa<c'f<c>'a4<cf' u-10 f'a4<c' u
 77:    'dgb' dg 'g4.b<d' u-20 'g4b' u 'fa<c'fa<'c*120f'>
 78:    'e4.gb' u-15 'b4.<e''g4b' u  'fa<c'fa<'c4.f'>'d4f'>
 79:    'b4.<ef' u-20 'b*120<ef' u 'a4.<cf''a*120<cf'<
 80:    l4. 'fb<d 'u-20 'b<df' 'f4b<d' u
 81:    u-10 <c8>f8a8 u-15 'a<cf''f4a<c'u
 82:    'gb<d' u-15 'd*120gb' u>'a<cf'< u-20f*120u
 83:    <g2. u-10 b4 >'b8<df' b8<f8 u-10 'd4fb' u-10
 84:    >b8<'d4fb' u > u-15 'c4.e' 'e*120fb'
 85:    'dfb' 'cfa'>'b4<eg' u-15 'b2<df' u l8<dc>b4
 86:    a<fa< u-30 'c4f'>fa u-10 g u
 87:    gbg 'b4<e' g u-20 b4 u
 88:    <d u-15 >fb u <d4 u-20 >f<c4> u
 89:    'dgb' u-20 dg u+10 'd4gb' u-30 fg4 u
 90:    <e> u-20 g<c u e4 >g< u-10 'c4e'> u
 91:    'a<cf' fa 'f4a<c' fa u-30 g u
 92:    'dgb' u-20 d'dg' u 'd4gb'db4
 93:    'g4.b<d''f4.b<c' u-20 'g4b<d' u
 94:    'gb<e'>gb<e4 gb< u-20 e > u
 95:    'a<cf' u-20 fa u 'a4<cf' u-30<cg4>u
 96:    'f4.b<d' u-25 'b4<d' u+10 'fa<c''f4b<d' u
 97:    'd4.gb' u-20 b4g<d4> u
 98:    'f!4.g<c' u-20 'f!4.g<c' f! u+10 f u+10
 99:    'cf!g'>g<cf!4 > u+15 g < u c4
100:    'e4.gb''e4gb' u-15 e>bg< 'e2gb' u 'f2a<c'
101:    'fb<d'f<d> 'b4<df' u-15 f<f4> u
102:    'fa<c' f<c> 'a4<cf' u-10 f'a4<c' u
103:    'dgb' dg 'd4g' bgd 'a4.<cf''f*120a<d'
104:    'egb' u-20 eg u 'g4b<e' u-20 gb4 u
105:    'd4.fb' u-15 'd*120fb' u
106:    >'g4.<ce' 'g4<ce' b<c>b
107:    'a4<cf' <f u-20 c4 u c> u-15 af< u
108:    'fb<d'f<d> 'b4<df' u-15 f'b4<d' u
109:    u-25 f*3  u g*21 u-20 af u 'a4<cf' f 'f4a<c'
110:    'g4.b<d' u-20 'g4b<d' 'b4.<d' u 'b1<d' <
111:    d4.'d*120fb' 'df' u-20 >b<d 'd4fb' u-10>b<'d4fb'>u
112:     u-30 'b4.<ce' u 'e4fb' 'efb'ef
113:    'd*120fb' f<d>b 'f2.a<c' 'f4a<c'
114:    'd2.gb' 'd4gb' 'fa<c'cf'f*120a<c'
115:    'd*120fb' f<d>b 'f2.a<c' 'f4a<c'
116:    'd*120gb' u-25 bg4 u+15 f u+10a u-20f4 u g4a4
117:    'fb<d'120 f<d>b 'fa<c' u-15 fa u-10 'a<cf'120 u
118:    'g4b<d' g 'gb<d'120 f<c>a 'a<c'120
119:    beg 'g4b<e' u-20 'gb' gb u 'dfb' cd 'd4fb' bf4
120:    u+20'c4.eg'  u 'c4eg' gc4
121:    u+20 >'a4.<cf'u 'a4<cf'<c>a u-20 f u <
122:    'fb<d' u-25f u+10b u 'f4b<d'f'f4b<d'
123:    g*4a*20gf 'a*120<cf' bdg 'g*120b<d'
124:    'd4.fa' u-20 'd*120fa' u
125:    'b2.<eg' <b4> 'b4.<df' < 'd*120fb' >
126:    r*312 u-20b u <b>u-10b u
127:    <e>b<c > 'b4<ce' 'g4.b' 'c1fa'
128:     |:4 |3¥15:|
129:    r2. 'f4b<d' 'f2a<c' g&u-40 a4. u
130:    'g1.b<d' r8 f'f<d'f'f2.<d' 'f4b<d'
131:    'b2.<d' 'd4f' 'c2.eg' 'e4g'
132:    'd2.fb' 'f4b<d' 'f4.a''fa'120
133:    'g1b<d' 'a4.<cf' 'fa<d'120
134:    |:'f2.b<d''f4b<d':| 'g1b<e'  :|
135:
136: /------------------------E T C .
137: (t2)       o3 l2 q8 i0@1
138:    'a<cf'< u-20 c4.c8>  'a<cf' u-20 <c4.c8> u+40
139:    'fa<c' u-15 <c4.>a8 u-20 'gb<e'<e4.>e8 dc>bg u+35
140:    <c>b u-10 a u-20 a4.>g8 u+15 g<ggg4.>e8<
141:    u+15 'gb<e'e4.>f8< 'a<cf' g4 u+15 a4
142:    bagf u+15 f! u-20 f! u+15 'gf!' f! r1r1
143:    @e100,20 r*1344 r2.
144:    l16 o3 @47 @v110 @u90  b<cde u110f1u
145:    l1 rrr2 @49 l4dfg2r2 r1r1r1r1
146:    @y1,102,80
147:    r*5184 r2. l1 q5f4q6g2r2
148:    u+15 i127@49 rr< d2.r4  u125 @v125
149:    >>rrr l4r b8<c8 de8f4.<c2>a8f8eb.
150:    e16d16c8>b8<c1
151:    l1 r*2304
152:    i0@101 u70 |:gf:|
153:
154: /------------------------V O C A L
155: (t3)       o5 l8 q8 i0@72 @m20 @h40
156:    r1r1
157:    r4 fff u-30c4. u r4 e4>b4< c u-35e4u r
158:    dedc4> b u-20b2.< u cd q7|:4e:|q8 e4.
159:    ed4c u-25c4. u >b<c d4. ed4c>u-20b&b1 u<
160:    r4 ggg b*120 ag u-15f4 u de l4ffdc>b2.< l8
161:    >gb<d4q7cq8c4.> gb<d4q7cq8c4>gb4< r4
162:    dcc4 u-30>b<uc2 u+10 fguf u+10f4 @u112
163:      |: >b4.b4<cd4 f4.f4ga4 r |:4b:| ag4f1 |
164:    bb4b4ag4 ff4f4ed4 r4 c>b<cd4 u-30c u c4 f4gff4 :|
165:    l4 q5<d.>b.q8gfg8>b l8 gb4< r4 ed4>bg4b1 <r1
166:    r4 fffc4. r4 e4>b4< u-10c u-20e4 r u
167:    dedc4> b u-20b2.u < cdee4.e4dc&c2. >b<c
168:    d4.e4dc>b&b1< r4 ggg4.ba4gf4. def4ffd4c>b&b2.<
169:    >gb<d4q7cq8c4.> gb<d4q7cq8c4>gb4<
170:    r4 dcc4 u-10>b<u-15c2u fgu-25fuf4
171:      |: >b4.b4<cd4 f4.f4ga4 r4 |:3b:| agff1 |
172:    bb4b4ag4 ff4f4ed4 r4c>b<cd4cc4 f4gff4:|
173:    l4 q5<d.>b.q8gfg8>b l8 gb4< r4 ed>b4g4b1 <r1
174:    r*1008
175:    q6f4q8gff4
176:    >b4.b4<cd4 f4.f4ga4 r |:4b:| ag4f1 |
177:    bbbb4ag4 ffff4ed4 r4 c>b<cd4 u-20c u c4f4gff4
178:    >b4.b4<cd4 f4.f4ga4 r4 b4bagff1
179:    <q5d4.>b4.q8g4f4b b&b2
180:    r1r2r
181:    > gb4 <e2d2>b2.g4
182:    |:6|5¥15:|
183:          b*120<cd4 f2<c4.>bb*120agff1
184:    <d4.>b4.g4f4 bb*120 r4 ed>b4 u-20g4u :|
185:
186: /------------------------V O C A L 2
187: (t4)       o5 l8 q8 i0@73 @m20 @h40 @k3
188:    r1r1
189:    r4 fff u-30c4. u r4 e4>b4< c u-35e4u r
190:    dedc4> b u-20b2.< u cd q7|:4e:|q8 e4.
191:    ed4c u-25c4. u >b<c d4. ed4c>u-20b&b1 u<
192:    r4 ggg b*120 ag u-15f4 u de l4ffdc>b2.< l8
193:    >gb<d4q7cq8c4.> gb<d4q7cq8c4>gb4< r4
194:    dcc4 u-30>b<uc2 u+10 fguf u+10f4
```

```
195:     @k0
196:       |: >b4.b4ab4a4.a4b<c4 r |:4d:| c>baa1 |<
197:     gg4g4fe4 dd4d4c>b4 r4 agab4aa4 <c4>a4f4:|
198:     l4 <q5g.f.q8ede8d l8 >gb4< r4 gg4de4d1 r1
199:     @k3
200:     r4 fffc4. r4 e4>b4< u-10c u-20e4 r u
201:     dedc4> b u-20b2.u < cdee4.e4dc&c2. >b<c
202:     d4.e4dc>b&b1< r4 ggg4.ba4gf4. def4ffd4c>b&b2.<
203:     >gb<d4q7cq8c4.> gb<d4q7cq8c4>gb4<
204:     r4 dcc4 u-10>b<u-15c2u fgu-25fuf4
205:             @k0
206:       |: >b4.b4 ab4 a4.a4b<c4 r4 |:3d:| c>baa1 |<
207:     gg4g4fe4 dd4d4c>b4 r4agab4aa4 <c4>a4f4:|
208:     l4 <q5g.f.q8ede8d l8 >gb4< r4 ggd4e4d1 r1
209:     r*1008
210:     q6f4q8gff4
211:     >b4.b4 ab4 a4.a4b<c4 r |:4d:| c>b4a1 |<
212:     gggg4fe4 dddd4c>b4 r4 agab4 u-20a u a4<c4>a4f4<
213:     >b4.b4 ab4 a4.a4b<c4 r4 d4dc>baa1
214:     <q5g4.f4.q8e4d4d d&d2
215:     r1r2r
216:     > gb4 <g2f2 d2.e4
217:     |:6|5¥15:|
218:             d*120 cd4 f2fga4g*120fe4d1
219:     b4.g4.e4d4 dd*120 r4 gf d4 u-20e4u :|
220:
221: /------------------------B A S S
222: (t5)        o3 l4 q8 i0@33
223:     r*3456
224:     l4 >b.<f.>b8f8 a.<f.>a g.<d*120>d.a.<d
225:     u-30d*6ue*66 >b.e d.b.dc.g.<c> f.f8ga
226:     b.<f.>ba.<f.>ag.<d.>gd.a.<d u-30d*6ue*186
227:     >d1c2f2 (ab)l8&a*366
228:     l1 fe>b2.a4g <c2>b2a g2.<g8d8>g2.d8 g8<
229:     l4 e.e8e2 f.f8f2 >b.b8a2 g.g8f2
230:     f!.f!8f!2 <c.c8c2 f.f8f2 >f.f8f2
231:     b.<ff8b >a.<f.>a g.<d.g d.a.q5dq8
232:     e.b.e8>b8<d.b.d c.g.d8e8f2>f2
233:     b.<f.>ba.<f.>a g.<d.g d.a.d
234:     e1d1 c2>f2
235:     l1 bbb  b2.f8&(f8a)
236:     bbb l4ffga
237:       |:    b.<f.>ba.<f.>a g.<d.g d.a.|d8>b8<
238:     e.b.e d.b.d c.g.c8>g8 f.f8ga :|
239:     <d |:e1d1:| c1>f1
240: |:6|5¥15:|b.<f.b8f8 >a.<f.>a g.<d.g d.a.<d8> u-20b8 u
241:     e.b.>b8<e8 d.b.d*28&(d*20>b)< l8cdef!fc>l4f:|
242:
243: /------------------------G U I T A R
244: (t6)        l8 o3 q3
245:     r*3456
246:       |: i8@29
247:     l8|: r2|:'dfb':|r'dfb'r2 |:'dfa' :|r4
248:     r2|:'dgb':|r'dgb'r2 |:'fa<c':|r4 |
249:     r2|:'egb':|r'egb'r2 |:'dfb' :|r4
250:     r2|:'eg<c':|r'eg<c'@u50r|:3'cfa'u+10 :|r'c4.fa' :|
251:     l1rrr| l8rrr
252:     << u+20 i8@28 q8 (df)6&d*66e4d1 @u90
253:     r1r1 l8 r*168 z100,,70,55,45,30 f4c4>b4<fc>b<
254:     r1r1    r*120
255:     @u100 q8 i8@28  z60,110,70,100,70,90
256:     >(b4<d),30 b4dad (g4a),l8z60g4
257:     l1rr l8rrr @u110 f4 z80,65cd4
258:     z,110,75,60,60,50,45 >b<fd>b4<fd>b z
259:     l1rrrr @u80 :| i8@28
260:     l4 <|:z60,50,100,80,60 c>b8<fd>b8< c>a8<fc>a8<
261:       e>b8<g(e4.c),48 |z80,70,110c>a8<f*120:|>
262:     r2 @u90 <f2> @u80 i8@29q3>
263:      l8|:r2|:'dfb':|r'dfb'r2 |:'dfa' :|r4
264:     r2|:'dgb':|r'dgb'r2 |:'fa<c':|r4 |
265:     r2|:'egb':|r'egb'r2 |:'dfb' :|r4
266:     r2|:'eg<c':|r'eg<c' i8@28 q8
267:     <a4<u+20au-20f4fc>u-10f>u q3 i8@29 :| @u90
268:     r1r1 i0@26 l4 q8<<c*4d*68>b.gf.b*120
269:     r1 b<e8f u-10e8u-20cu l4 i8@28
270:       |:6|5¥15:|
271:     z60,65,100,70,50 c>b8<'fd'd8>b< c>a8<'cf'c8>a<
272:     d>b8<'dg'd8>b< fc8'fa'f8c
273:     l8 z60,,,100,70,55 f>b<dg4d>b4< f>b<d'f4b'df4
274:     z70,100,55 c4'gb'c z70,100,60,50c'gb'ec :|
275:
276: /------------------------G U I T A R 2
277: (t7)        i8@28 o3 q8  l1 r*7 @k14
278:     r*3456
279:       |:    |:r*768 |r*768:|
280:       r*576 |l8rrr
281:     << u+10  (df)6&d*66e4d1
282:     r1r1 l8 r*168 z90,,70,55,45,30 f4c4>b4<fc>b<
283:     r1r1    r*120
284:     @u100  z60,100,70,100,60,80
285:     >(b4<d),30 b4dad (g4a),l8z60g4
286:     r1r1 l8rrr @u100 f4 z80,65cd4
287:     z,100,70,60,55,45,40 >b<fd>b4<fd>b z
288:     l1rrrr :|
289:     l4 <|:z60,50,100,70,50 c>b8<fd>b8< c>a8<fc>a8<
290:       e>b8<g(e4.c),48 |z80,70,100c>a8<f*120:|>
291:     r2 <f2> >
292:         |:r*768|r*576 l8<a4<u+20au-20f4fc>u-10f>u :|
293:     r1r1 @u75 i0@26 l4 <<c*4d*68>b.gf.b*120
294:     r1 b<e8f u-10e8u-20cu l4
295:       |:6|5¥15:|
296:     z60,65,100,70,50 c>b8<fd8>b< c>a8<cc8>a<
297:     d>b8<dd8>b< fc8ff8c
298:     l8 z60,,,100,70,55 f>b<dg4d>b4< f>b<df4df4
299:     z70,100,55 c4gc z70,100,60,50cgec :|
300:
301: /------------------------D R U M
302: (t9)        i0@1 q8 o2 l4
303:     r*3456
304:     <u+20g2u>r|:4f:|f8a8
305:     |:7f:|f8f8 |:15f:|f8a8 |:15f:|r
306:     l2  u-10  <rdrd u-20d dd z80,45,60,100 d*72d16d16d1 u
307:     l4 u+20g2u>r|:10f:|f8f8ff |:15f:|f8a8
308:     |:21f:|f8f8ff |:6f:| rr
309:     |:<u+20g2u>r|:12f:||f8a8   |:14f:|rr :|
310:     u-10 <r2. |:3 u-10|d2:|d4. u
311:     u-30|:3d*8:| u d2d4 u-78|:6d*8u+13:|u d2
312:     o4l2 |:4a-| u-15 r4.a-8 u r2r2:|r1o2l4
313:     |: <u+20g2u>r|:5f:| |:7f:| |f8a8   |:14f:|rr :|
314:     u-10  r l2< |: dddd*72 u-45|:3d*8u+15:| :|
315:     ddl4d.dd8d8d8 u
316:      |:6|5¥15:|
317:     o3u+20g2u>r|:12f:|f8u-20f8u|:8f:| <du-20d8u r8r2 :|
318:
319: /------------------------D R U M 2
320: (t10)       i0@1 o2 q8
321:     r*3264 r2.f4
322:     l2 |:11 c4.c8 'c<f+o6c+' :|
323:     c4.l8c 'c<f+o6c+' u-15 bg4
324:     l1 rrrr r4. a4.f4 u
325:     l2|:15 c4.c8 'c<f+o6c+':| c4.l8cdaf4
326:     |:7 c4.c u+10 d2 u:| c4.cda4.
327:     |:3 c4.c u+10 d2 u:| c4.cd4a4
328:     l1 rrr l8
329:     |:3c*168c8c2f2:| c*168cc<c4.>bgf4
330:     l8
331:     |:7 c4.c u+10 d2 u:| c4.cdbgf
332:     |:3 c4.c u+10 d2 u:| c4.cd4a4
333:     r*960
334:     l8 rrr'b4<c'bgf
335:     |:   |:6 c4.c u+10 d2 u:| c4d<c4>agf
336:             |:6 c4.c u+10 d2 u:| c 'd4.<c',3 'd<c',1 agf
337:             |:6 c4.c u+10 d2 u:| cdddbagf
338:             |:6 c4.c u+10 d2 u:| cb4.ddg4 ¥15  :|
339:
340:
341:
342: /------------------------S T R I N G S 1
343: (t11)       i0@50 l1 o4
344:     r*4992
345:     b<cd r2 l4d @49u+10f u+15g4u r2. @50l1
346:     r*2112
347:     r4. >l8gab<cd e2.d4c*120>f<g4f*120cd4>b1
348:     g1^1 r1r1
349:  |:i0@50l1 b<cd l4 f2<c>f g2.bf2.b e2.<u-30cu> r2.
350:     l16 i127@49 fgaf l4 b2.f<c2.>f<d2.>g<f.f!8d
351:     u+20 > q5fq6g2 q8u r2 r1|r1
352:     l8 rrr  @v123 u+5c u+5d4
353:     u+5ef4.<c2>bab4<e4.dc>b<c>f
354:     <f4.edc d1u-10c1u > b*120 g16f16 edu-25c1u @v115 :|
355:     i0@101 u-30 >>g1f1< r1r1 @u115
356:      |:6|5¥15:|
357:     i0@50 l8 b2^8 @49 <cd4f2fga4 b2^8ag4f2r
358:     >{fgab<cdefgab<cdef}4.& f4.d4.>b4f2d2>b1 :|
359:
360:
361:
362: /------------------------S T R I N G S 2
363: (t12)       i0@50 l1 o3
364:     r*4992
365:     b<cd r4.f8d2 'egb' r 'ceg'>'b<df'384<
366:     r*1344
367:     l8 r*528 'gc4<e''fb<d'120
368:     'cfa' 'd4fb'> 'b1<df'< 'cf!g'384 r1r1
369:      |:      l1 >b<cd >
370:     'b2<df' < 'f4a<c' > 'b4<df' <
371:     'c2.eg'  'd4fb' >'b2.<df'< 'd4fb'>
372:     'g<ce' r1 r1r1r1u100 'a4.<df''a8<df!'
373:     'f2a<d' < 'dgb'| r 'ceg'
374:     l8 rrr @v110 u+15 cd4ef4.<c2>bab4<e4.dc>b<c>f
375:     u-5  'a4.<cf''gb<e''fb<d''fa<c' 'f1b<d'
376:          'f1a<c' 'egb'120 'c16eg'>'b16df'
377:          'b<ce''gb<d'    'f1a<c' u @v100:|
378:     @47 r*120 l8 u80{fgab<cdefgab<c}4.<ud1 >f4.b2^8
379:     r*1728 @u100
380:      |:5|4¥15:|
381:     o5i0@50 l8 b2^8 @49 <cd4f2fga4 b2^8ag4f2r2 r1r1r1:|
382:
383:
384: (p)
```

リスト2 LOVE IS ALLのカウンタ表示

```
 1:000064B0 00000000   2:000043B0 00000000   3:00005F70 00000000   4:00005F70 00000000
 5:00005F70 00000000   6:00005F70 00000000   7:00005F77 00000000   9:00005F70 00000000
10:00006F30 00000000  11:00005F70 00000000  12:00005F70 00000000
```

▶ 8月号の「餓狼伝説SPECIAL」の広告でジョー東は佐竹雅昭にしか見えない。
高畑　健治(18)兵庫県

リスト3 季節風

```
  1: .comment                    K I S E T S U - F U
  2: .comment             (HELLHOUND rd.3) (c)BREAK THROUGH 1994.
  3: .comment              By Y.komatsu /T.egawa /K.yoshida /K.T
  4: /  ～今 季節の風が 耳元すぎる かけぬけてけ Fairly Wind
  5: /                   伝えたい この胸の想い すべて～
  6: (i)
  7: (o175)
  8: (m1,2000)(afm5,1)
  9: (m2,2000)(afm2,2)
 10: (m3,2000)(afm3,3)
 11: (m4,2000)(afm4,4)
 12: (m5,2000)(afm1,5)
 13: (m6,2000)(afm6,6)
 14: (m9,2000)(aadpcm,9)
 15: .adpcm_block_data=kisetufu.zpd
 16: / BRASS
 17: /       Ar  Dr  Sr  rr  SL  OL  KS  ML DT1 DT2 AME
 18: (@1,    25,  5,  2,  6,  9, 27,  1,  1,  2,  0,  0
 19:         22,  6,  2,  7, 10,  0,  0,  1,  5,  0,  0
 20:         25,  3,  2,  6, 11, 33,  0,  1,  2,  0,  0
 21:         22,  6,  2,  6, 11,  0,  0,  1,  5,  0,  0
 22: /       AL  FB  OM PAN
 23:          4,  7, 15,  3)
 24: / BASS
 25: /   Ar D1r D2r  rr D1L  TL  rS MUL DT1 DT2
 26: (@2,     31, 15, 12,  6, 14, 25,  0,  2,  0,  0,  0
 27:     28, 13,  5,  7, 10,  0,  0,  0,  0,  0,  0
 28:     20, 13,  5,  7, 10,  0,  0,  1,  0,  0,  0
 29:     20, 14,  5,  7, 10,  0,  0,  2,  0,  0,  0
 30: /  CON FBL  SM VOL
 31:      5,  7, 15,  3)
 32: / STRINGS
 33: /   Ar D1r D2r  rr D1L  TL  rS MUL DT1 DT2
 34: (@3,     20,  8,  0,  2,  1, 25,  0,  1,  6,  0,  0
 35:     15,  5,  0,  7,  2, 25,  0,  4,  3,  0,  0
 36:     15,  5,  0,  7,  2, 30,  2,  3,  0,  0,  0
 37:     15,  5,  0,  7,  2,  0,  0,  2,  7,  0,  0
 38: /  CON FBL  SM VOL
 39:      5,  7, 15,  3)
 40: / BELL
 41: /   Ar D1r D2r  rr D1L  TL  rS MUL DT1 DT2
 42: (@4,     31, 19,  4,  5,  2, 27,  0, 11,  3,  0,  0
 43:          24, 17,  8,  6,  0,  0,  1,  1,  3,  0,  0
 44:          31, 19,  4,  5,  2, 27,  0, 11,  6,  0,  0
 45:          24, 17,  8,  6,  0,  0,  1,  1,  6,  0,  0
 46: /        AL  FB  OM PAN
 47:           4,  4, 15,  3)
 48:
 49: (t1)[do]q818 k1 v14o5@1 @h50@s5@m22                    /Melody
 50: (t6)[do]q818 k1 v11o4@1                                /Melody'
 51: (t2)[do]q718 k1 v10o5@4                                /Bell
 52: (t3)[do]q818 k1 v11o4@3                                /Strings
 53: (t4)[do]q818 k1 v12o4@3                                /Strings'
 54: (t5)[do]q818 k1 v14o4@2                                /Bass
 55:
 56: (t1) a2..b& b4.<c&c>b4a a2.bb& b4.arab4
 57: (t1)a4.b&b4<c4& c>b4.arba& a1 g4. a2&a8
 58: (t1)a4.b&b4<c4& c>b4.arba& a4.<d&d4.g& g2f+rgd&>
 59: (t1)<d2.&dd& dc>b<c&c>bag g2.aa& a4.b&b<cde>
 60: (t1)<f+4.g&g4.f+& f+f+grf+rga& ag4d&d4.~3d&dc>b<cd.e16&ef+>_3
 61: (t1)<f+4.g&g4.f+& f+f+grf+rgd& d1& d2..f+&>
 62: (t1)<f+4.g&g4.f+& f+f+grf+rga& ag4d&d4.c& c4>b4a4g+g&
 63: (t1)g1 g4f+gra4. g2..f+& f+4.e&ef+4g
 64:
 65: (t2)<d>da<d>da<d>d a<d>da<d>da<dd>da<d>da<d>da<d>dagf+ae
 66: (t2)<d>da<d>da<d>d a<d>da<d>da<d d>agd&dga<c& ccdrcrdd>
 67: (t2)<d>da<d>da<d>d a<d>da<d>da<d d>da<d>da<d>d gd<d>dgd<d>d
 68: (t2)<c>cg<c>cg<c>c g<c>cg<c>cg<c> <d>da<d>da<d>d >
 69:     l16 def+ab <cdef+ab<cc+def+> l8
 70: (t2)l8<f+d>ar4<c>gr <f+dgdf+dgr c+>ad<d&d>adf+ def+gab<de>
 71: (t2)r<d>ar4<d>ar<f+dgdf+dg>g l16 >a<df+af+d>a<df+af+d>a<df+a
 72:         <df+d>af+a<df+d>af+a<df+d>a
 73: (t2)cdf+a<c>af+dcdf+a<c>af+d cdf+a<c>af+dcdf+a<c>af+d
 74:     df+ab<d>baf+df+ab<d>baf+ fab<dfd>bafb<dfc+g+cg>
 75: (t2)a<cdegedc>a<cdegedc> a<cdegedc>a<cde>ega<c>
 76:     df+a<c>df+a<c>df+a<c>df+a<c> f+a<cd>a<cdf+cdf+adf+a<c>>
 77:
 78: (t3) d1& d1 d1& d1
 79: (t3)d1  ~2 d1 ~2    f+1 _4 c4. d2&d8
 80: (t3)d1 d1 d1 d2>g2<
 81: (t3) >g1& g2<~2d2~2 g1~2 a1
 82: (t3)_2r4dre4rd& dderdrer >ar4b&b2& b1<
 83: (t3)r4dre4rd& dderdr16err16 gr4f+&f+4.r er4f+&f+4.r
 84: (t3)r4dre4rd& dderdrer >ar4b&b2< fr4d&d2
 85: (t3)r4>gr<c4.r >g4.r<c&c4. r4drf+4.d& d4.f+&f+a4.
 86:
 87: (t4) f+1& f+1 f+1& f+1
 88: (t4)f+1 ~2 f+2 g2 ~2 a1 o4 @1g4.a2&a8 _4
 89: (t4)o4v12@3g1 g1 f+1 g2d4>b4<
 90: (t4)c1& c2~2g2~2 <c1~2 d1>
 91: (t4)_2r4f+rg4rf+ f+f+grf+rgr c+r4d&d2& d1
 92: (t4)r4f+rg4rf+& f+f+grf+r16 grr16 br4a&a4.r gr4a&a4.r
 93: (t4)r4f+rg4rf+&  f+f+grf+rgr c+r4d&d2 ar4g&g2
 94: (t4)r4crg4.r c4.rg&g4. r4f+ra4.f+& f+4.a&a<d4.>
 95:
 96: (t5)>c4cc&cc&cc c4cc&cc>b<c>b4bb&bb&bb b4bbbb<c>b<<
 97: (t5)>c4cc&cc&cc c4cc&cc>b<c d4dddddd d4ddddc+<
 98: (t5)>c4cc&cc&cc c4cc&cc>b<c>b4bb&bb<cd e4ee&eede<
 99: (t5)>>a4aa&aa&aa a4aa&aab<c d4dd&dd&dd d4dd&ddc+c+<
100: (t5)>c4.c4.c4 c4.c4>f+a<c>b4.b4.<c4>b4.b4f+ab<<
101: (t5)>c4.c4.c4 c4.c4ccc+ d4.d4dd&<r>dd>g<dd>g<dc+<
102: (t5)>c4.c4.c4 c4.c4>f+a<c>b4.b4b<cc+16d16 e4.e4ede<
103: (t5)>>a4.a4.a4 a4.a4ab<c d4.d4ddd< d4ddddc+c+
104:
105: (t6)
106:      d*0&f+*0&@q28a2..  d*0&g*0&b8&      d*0&g*0&@q16b4.
107:      e*0&g*0&@q8<c4>    d*0&g*0&b4       d*0&f+*0&@q4a8
108:      d*0&f+*0&@q24a2.   d*0&g*0&@q4b8    d*0&g*0&b8&
109:      d*0&g*0&@q16b4.    d*0&f+*0&@q8a4   d*0&f+*0&@q4a8
110:      d*0&g*0&@q8b4      d*0&f+*0&@q12a4. d*0&g*0&b4.
111:      e*0&g*0&<c4&>      e*0&g*0&<c8>     d*0&g*0&b4.
112:      d*0&f+*0&@q8a4     d*0&g*0&@q4b8    d*0&f+*0&a8&
113:      d*0&f+*0&@q36a1    c*0&e*0&@q12g4.  d*0&f+*0&a2&
114:      d*0&f+*0&@q20a8q8  d*0&f+*0&a4.     d*0&g*0&@q12b4.
115:      e*0&g*0&<c4&>      e*0&g*0&<c8>     d*0&g*0&b4.
116:      d*0&f+*0&@q8a4     d*0&g*0&@q4b8    d*0&f+*0&a8&
117:      d*0&f+*0&@q16a4.   f+*0&a*0&<d2>    b*0&<e*0&g8&>
118:      b*0&<e*0&@q20g2>   a*0&<d*0&@q8f+4> b*0&<e*0&@q4g8>
119:      e*0&a*0&<d8&>      e*0&a*0&@q32<d2..>e*0&a*0&<d8&>
120:      e*0&a*0&<d12&>   e*0&a24& e*0&a*0&<c12^48&> e*0&a48
121:      e*0&a*0&b12^48&  e*0&a48  e*0&a*0&<c6^24&>  e*0&a24
122:      e*0&a*0&b12^48     e*0&a48& e*0&q4a8@q4g8
123:      c*0&e*0&@q24g2.    d*0&f+*0&@q4a8   d*0&f+*0&a8&
124:      d*0&f+*0&@q40a2    d*0&f+*0&a6^24&d24&
125:      'f8d8g+'&r12q4c+24>a*0&<d*0&@q12f+4.>
126:      b*0&<d*0&@q16g2>   a*0&<d*0&f+8&>   a*0&<d*0&@q8f+8>
127:      a*0&<d*0&@q4f+8>   b*0&<d*0&@q8g4>  a*0&<d*0&f+4>
128:      b*0&<d*0&@q4g8     c+*0&f+*0&a8&    c+*0&f+*0&@q8a8>
129:      b*0&<c+*0&g4>      f+*0&b*0&<d2^8&> f+*0&b*0&@q52<d1>
130:      a*0&<d*0&@q12f+4.> b*0&<d*0&@q16g2> a*0&<d*0&f+8&>
131:      a*0&<d*0&@q8f+8>   a*0&<d*0&@q4f+8> b*0&<d*0&@q8g4>
132:      a*0&<d*0&f+4>      b*0&<d*0&@q4g8>  f+*0&a*0&<d8&>
133:      f+*0&a*0&<d1&>     f+*0&a*0&@q64<d2..> a*0&<d*0&f+8&>
134:      a*0&<d*0&@q16f+4.> b*0&<d*0&g2>     a*0&<d*0&f+8&>
135:      a*0&<d*0&@q8f+8>   a*0&<d*0&@q4f+8> b*0&<d*0&@q8g4>
136:      a*0&<d*0&f+4>      b*0&<d*0&@q4g8   c+*0&e*0&a8&
137:      c+*0&e*0&@q8a8>    b*0&<c+*0&g4>    f+*0&b*0&@q16<d2
138:      f*0&a*0&<c8&>      f*0&a*0&q6<c4>   f*0&g+*0&@q8b4
139:      d*0&f*0&a4         d*0&>b*0&@q4<g+8 c*0&e*0&g8&
140:      c*0&e*0&@q36g1     c*0&e*0&@q8g4>   a*0&<d*0&@q4f+8
141:      c*0&e*0&@q8g4 d*0&f+*0&@q12a4.>     a*0&<d*0&@q28g2..>
142:      a*0&<d*0&f+8&>     a*0&<d*0&@q16f+4.> a*0&<c*0&@q8e4>
143:      a*0&<d*0&f+4       c*0&e*0&@q4g8
144: (t9)o6l8 d4
145:  [do]
146:  o6 e d4 d e c+& |:3  dfed4ded d4ed4dec+& :|
147:  o1  b4<d<c6r24>f+24ad12.f12.<g+4&>> o6  ed4dec+&
148:  |:3dfed4ded d4ed4dec+&:| o1 b<d4d4ddd<g+4&>> o6 ed4dec+&
149:  |:3dfed4ded d4ed4dec+&:| o1 bb<d>b<f+ad>b<<g+4>> o6 ed4dec+&
150:  |:2dfed4ded d4ed4dec+&:| dfed4ded
151:  o1 <<g+4>d2>b4<<<<< g4.>>> c6r24>f+24ad12.f12.<g+4&>>
152:  [loop]
153:
154: (t1)[loop]
155: (t2)[loop]
156: (t4)[loop]
157: (t3)[loop]
158: (t5)[loop]
159: (t6)[loop]
160:
161: (p)
```

リスト4 季節風の音色コンフィグファイル

```
.o1b  = drk2.pcm,v62
.o2d  = extra_perc¥flgsn.pcm,v121,c0,2200,f1600,0
1=extra_perc¥gttom2.pcm,c0,3400,f2000,0,p-3,v73
.o2f  = 1,p-6,c0,3400,f2000,0
.o2g  = 1,p-3,c0,3400,f2000,0
.o2a  = 1
.o2b  = 1,p3
.o3c  = 1,p7
.o3d  = 1,p11
.o3c+ = extra_perc¥crash_2.pcm,c0,8800,v85,f111,25
```

```
.o3g+ = .o3c+,mo1b
1 = extra_perc¥hhato.pcm v70
.o6c+=1,mo1b
1 = extra_perc¥hihat_c.pcm,f-100,80,p-1 v210
.o6d=1,mo1b
.o6e=1,mo2d
.o6f=1
.o6g = .o2d,mo3c+
.erase 1
```

リスト5 季節風のカウンタ表示

```
1:00000000 00001800  2:00000000 00001800  3:00000000 00001800  4:00000000 00001800
5:00000000 00001800  6:00000000 00001800  9:00000030 00001800
```

▶最近CMで自分の名前を呼ばれている。思わず返事をしたくなる。
伊藤　充(タカシ)(20)神奈川県

リスト6　踏切の通過音

```
 1: .comment 踏切（電車内から） programmed by M.Hasunuma
 2: (i)
 3:
 4: .adpcm_block_data = tr
 5:
 6: (m1,1000)(afm1,1)
 7: (m2,1000)(afm3,2)
 8: (m3,1000)(aadpcm,3)
 9:
10: /        AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME
11: (@1,     28,  4,  0,  5,  1, 37,  2,  1,  7,  0,  0
12:          22,  9,  1,  2,  1, 47,  2, 12,  0,  0,  0
13:          29,  4,  3,  6,  1, 37,  1,  3,  3,  0,  0
14:          15,  7,  0,  5, 10,  0,  2,  1,  0,  0,  1
15: /        AL  FB
16:           2,  7)
17:
18: (o120)
19:
20: (t1)     @1p1@v60o5@k421*44      /警報機（左ch）
21:          |:10'ff+'¯2:|
22:          |:10'ff+'¯3:|
23:          @k42,-42'f*46f+'@k-421*48|:10'ff+'_7:||:10'ff+'_2:|
24:
25: (t2)     @1p2@v20o5@k421*44      /警報機（右ch）
26:          |:10'ff+'¯2:|
27:          |:10'ff+'¯7:|
28:          @k42,-42'f*46f+'@k-421*48|:10'ff+'_3:||:10'ff+'_2:|
29:
30: (t3)     o4|:32c*72:|            /電車
31:
32: (p)
```

リスト7　踏切の通過音の音色コンフィグファイル

```
.o4c = tr.pcm,p+6
```

リスト8　踏切の通過音のZPDデータ制作用バッチファイル

```
zplk -x,,1 train_lp.p16 tr.pcm
zpcnv tr
del tr.pcm
```

リスト9　踏切の通過音のカウンタ表示

```
1:0000075E 00000000   2:0000075E 00000000   3:00000900 00000000
```

◆LOVE IS ALL

ピアノが命。ベロシティで音色が変化するので，最適な強さを探してみてください。演奏が硬いところは，要所要所でダンパーを使うのもいいかも。あと発音数の関係か，途中から左手がなくなっているのが惜しい。全体的に音が中央付近に張りついた印象があります。パンとエフェクタの設定を見直すだけでも，段違いによくなるでしょう。危ない場所もそれほどなくて（ピアノが少し危ないけど），耳コピは合格点。それにしてもいい曲ですねぇ。

◆季節風

どこかで聴いたことあると思ったら，あのシューティングゲームの曲なんですね。パート構成などコナミの影響があるようですが，ブラス系の音が中心なので発音数の割に少々薄い気もします。「いかにも！」という曲ですが，淡々とした感もありますね。何かキャッチーな要素がひとつあればいいと思うんですが。

◆踏切の通過音

これは一発ネタのようですが，実は深いです。もうちょっと「高速で通過した」雰囲気が出ていればいいかもね。

(進)の「ちょっといいですかぁ?」

突然やってくるSC-55テクニック講座。今月はモジュレーションパラメータに変更を加えて「コントロールチェンジ1番（拡張モードでないときの@m）をさらに有効に」使う方法を紹介しましょう（SC-55ばかりだと怒らないでね）。拡張ピッチモジュレーション，拡張ARCCで音量モジュレーションやワウもどき……，これって実は，音源にとってあまり嬉しいものじゃないんです。とめどなく信号が送られてくるわけだから，多用すると比較的高速なSC-55でもテンポがヨレたりします（これは致命的）。

そこで，SC-55内蔵のモジュレーション機能に注目してみることにしましょう。

普通にマニュアルを眺めた限りではレイトとデプスくらいの変更しか見当たらず，何だか融通がきかない感じもします。拡張モードへ走ってしまうのも無理ないかもしれません。

では，SC-55マニュアルのいちばん後ろを見てください。パラメータアドレスマップ$40, $2n, $00番地から，MODなんちゃらというブロックがあります。コントロールチェンジ1番は，ここに設定された値を見て動作します（TONE MODIFY設定も関係しますが，基本はココ）。

パラメータ範囲とデフォルト数値を見れば，上から3つの項目では初期値が中点にあることがわかります。この3つは内部的に@mが有効であるとき，ピッチ，カットオフ周波数，音量を変化させるものですから，通常は中点で当たり前。楽器のジョイスティックがモジュレーションに対応していれば，角度によって状態を変化させるなどという用途に使えます。

続く8つのパラメータが重要。ここでは，震えの状態や効果を決定します。上からRATE（震えの周波数），PITCH DEPTH（音程震えの深さ），TVF DEPTH（フィルタ震えの深さ），TVA DEPTH（音量震えの深さ）とあります。ここをいじれば@mで音量モジュレーションやフィルタの周期的開閉などを，音源側で処理してそうです。しかも2セット用意されています。これはモジュレーションで2つのLFOを制御できることを意味します。初期値はPITCH DEPTHだけ64です。これではLFO1によりピッチが変化するだけですが，設定により音量やカットオフのコントロールも可能なわけです。2系統のLFOを使えば複雑な波形も作れますね。独立して別々の効果に使ってもいいでしょう（音程LFOを速く・音量LFOはゆっくりと，など）。

ちょっと話はそれますが，これら8つを全部0にするとLFOはかかりませんね。そこで，先ほど解説した先頭3つのパラメータだけを設定しておけば，@mでピッチその他を増減することもできます。@mを限定的にカットオフのコントローラにしてしまえば，ZAMやMONではその数値が確認できますね（NRPN設定でカットオフを操作しても見えない）。モジュレーションが使えなくなってしまいますが，そういった場合には汎用コントローラを使うという手もあります（CC1，CC2と2つ用意されています。解説はいずれまたということで）。

では具体的に設定してみます。

```
/ part = 1
 x$40,$21,0, 64, 60, 64 / PCH,TVF,AMP
 x$40,$21,3, 35,  0, 80, 10 / MOD LFO1
 x$40,$21,7,100, 20,120,  0 / MOD LFO2
 @m100          / 拡張モードにしちゃダメよ
 @h48 c * 384
```

音色やボリュームなどは適当に設定してください。あと，エクスクルーシブ送信でのアドレスが連続なのに3行に分けてあるのは，誌面の都合です。よほどのことがない限り，普通は全部繋げて記述しましょう。曲の途中だったりする場合はなおさらです（データ量の少ない方法で送信するほうがいいのはいうまでもないから）。

ローランドエクスクルーシブ送信なのでアドレスから書いてありますが，これはパートによって異なります。当然，正しく設定しなければダメです。これはパート1の設定例なのでご注意を。パート2の場合，MOD～に関わるアドレスは$40, $22, $00から始まります。

上の例では，モジュレーション動作中はカットオフ周波数が若干下がり，音程が速く，音量がゆっくりとウネウネします。フィルタは2つのLFOを合成したちょっと複雑な変化をしています（ワウワウとうるさいけど）。波形は1種類だけど，覚えておいて損はないと思います。

◆　◆　◆

このようなちょっとした手続きで，意外と演奏の可能性が広がったりします。基本的なことさえ知っていればある程度の音は出せますが，せっかく手に入れた音源，SC-55程度ならパラメータも少ないので突っ込んで勉強してみるのもいいでしょう。使えるパラメータを把握するだけなら難しくはありません。　（進藤慶到）

(善)のゲームミュージックでバビンチョ

西川善司

●CGMV・デイトナUSA
VHS:TYVY-5002　　4,900円(税込)
東芝EMI　　発売中

セガ「デイトナUSA」を動かしているシステムは，リアルタイムにテクスチャマッピング(立体に平面を張り付ける)，スムーズシェーディング(色の変化を自然にしたり面と面の繋ぎ目をわかりにくくすること)を実現でき，それまで折り紙のようだった3Dグラフィックに対して見事な質感の演出に成功している。以前のカーレースゲームとは段違いの視覚的リアリティを体感できる。

サウンドのほうも，もうほとんどバンド生演奏級の歌うBGM。効果音はレース状況に応じた簡単な実況中継までしてくれちゃう徹底ぶり。

グラフィックもここまでやるか，といわせてくれるくらい徹底している。コース周辺の海や建造物，さらに動植物などの「自然環境」までもさりげなく表現しちゃって(ちゃんと動いてたりするし)，さらには車のガラスに映り込んだ空や雲などもゆっくり動いてたりして，もはや芸術の域に達している。

さてビデオの内容はというと，突然「デイトナUSA」の謎の男2名によるレース実況中継から始まる。引き続き，ファンにはうれしい初級・中級・上級それぞれのコースの徹底攻略。攻略は各コーナーの理想進入速度，理想進入角度をさまざまな視点の映像を駆使してレクチャーしてくれる。

そして，ビデオのエンディングタイトルが出てもまだ巻き戻すな！　最後にとんでもない映像が収録されているぞ。マジで。

ゲームがバーチャルリアリティに向かって確実な進歩を果たしているのがなんとなくわかっちゃうって感じのビデオだね。

お勧め度　9

●國府田マリ子のRadio Canvas VOL.1
CD:KICA-7643　　2,800円(税込)
キングレコード　　発売中

一連の「ツインビーパラダイス」シリーズの番外編。ツインビーたちの住むどんぶり島のFMラジオ「db-FM」。このラジオ局で國府田マリ子が番組をスタート。楽しいお便りの紹介，幻の「ステレオドラマツインビーパラダイス」の第25話の特別放送，と盛りだくさんのプログラム。しかしここはどんぶり島，はたして誰にも邪魔されず國府田マリ子は番組を終了できるのだろうか。それは聴いてからのお楽しみ。

ところで，いつの間にこの國府田マリ子ってこんなに人気が出たの？　「ウインビー国民的アイドル化計画」はどうした？　このままだと國府田マリ子のほうがウインビーより人気出ちゃうびー。どうするびー。

お勧め度　7

●ぽっぷるメイル　サウンドボックス'94
CD:KICA-1148～1149　　3,600円(税込)
キングレコード　　8/24発売

1992年に日本ファルコムから登場したコミカルアクションRPG「ぽっぷるメイル」が家庭用ゲーム機に移植され再登場。BGMも各家庭用ゲーム機のハードに合った形にリニューアルされ生まれ変わっている。

2枚組構成で収録曲はDISC 1がSFC版のオリジナルサウンド全28曲，DISC 2にはメガCD版全19曲とPCエンジン版全17曲，そしてボーカルアレンジバージョン1曲が収録されている。

最近の日本ファルコムのゲームミュージックは質が高い。ラヴェル系や久石譲系の曲もあったりして，ちょっとニヤリとさせられる部分もあり。私は「バックアップ作成」の曲が気にいった。

お勧め度　8

●ウィンビーのネオ・シネマ倶楽部2
～パラダイス編～
CD:KICA-7642　　3,000円(税込)
キングレコード　　発売中

懐かしのコナミのゲームミュージックを爽やかに復刻アレンジ。収録タイトルは「ツインビー」「魍魎戦記MADARA 2」「エキサイティングサッカー」「悪魔城伝説」「グラディウス2」「月風魔伝」「XEXEX」「スナッチャー」「愛戦士ニコル」「グラディウス」の全部で10タイトル。

原曲のモチーフのおいしいところを巧みに使ってまったく別の曲を作ったというイメージが正しいかも。「夏」を思わせるアレンジの曲が多く，非常に聴き心地がよい。勉強のBGMに，ドライブのBGMに，そしてお休みのBGMに最適だ。

私のお勧めはトラック1「ツインビー」の「OPEN YOUR HEART」とトラック3「エキサイティングサッカー」の「MIRAGE」，そしてトラック10の「グラディウス」「BLUE DESERT」かな。

このCDの収録曲は，秋より再スタートするラジオドラマ「ツインビー」のパート2用のBGMとしても使われる。うーむ，ツインビーに対するコナミレーベルのこの意気込みはいったいどこからくるのだろう。

お勧め度　9

おわりに

カプコンの「ストリートファイターII」の実写版劇場映画がアメリカからやってくる。その写真資料を見たのだが，ケンはただの白人金髪少年だし，本田はハワイアンだし，キャミィはボンドガールだし，ブランカはただの好青年(怒ると変身するという設定らしい)，サガットはどう見ても単なる片目の中年ハゲ親父だし，ウワーなんだこりゃ，って感じ。さすがアメリカ，主人公はガイルだわい。私としちゃ，インドでダルシム主人公の映画を撮ってほしかったわーん。

ぽっぷるメイル　サウンドボックス'94

ウィンビーのネオ・シネマ倶楽部2

X68000用CARDDRV対応カードゲーム

ひとりポーカー

Furuki Kenichi 古木 健一

別名“ポーカースクエア”とも呼ばれるカードゲームです。カードを縦横5×5の25枚並べて，ポーカーの役を作っていきます。あなたはひらめきで勝負しますか，それとも確率勝負？ 設定も変えられますのでいろいろ試してみてください。

入力方法

このゲームはCARD.FNCシステムに対応したカードゲームです。

CARD.FNCをお使いの方は，CARD.FNCを組み込んだBASICを立ち上げるか，CARDDRV.XでTR.DATを登録してCARD2.FNCを組み込んだBASICを立ち上げてリストを入力してください。

ゲームの説明

このゲームは基本的にポーカーです。よく切った52枚のカード（ここではジョーカーも使用可）を1枚ずつめくり，25枚のカードを縦5×横5枚になるように場に並べていきます。そして，縦，横，斜めのラインにポーカーの役ができていれば，その役にしたがって得点が入ります。

プログラムを実行すると選択肢が現れますので，とりあえず開始を選んでください。開始を選ぶとカードが切られて，画面右下に手札の山が置かれます。次に，トップカードがマウスカーソルに移動するので，カードを置きたい場所で左クリックしてください。これを繰り返していき，場がカードで埋まるとゲーム終了です。

ちなみに，一度置いたカードは動かすことはできませんし，カードを重ねて置くことも禁止されています(当然ですよね)。

おまけ機能

先ほど，開始を選んだときにいくつかの項目があったはずです。それについて解説しておきましょう。

●ジョーカーの数(0〜2)

ジョーカーの枚数を指定できます。ルールを参考にした本ではジョーカーを除く52枚のカードを使うと書いてありましたが，得点を競うからには役ができにくいとつまらないので，ジョーカーを2枚まで使えるようにしました。このため，オリジナルにはない役「ファイブカード」を作ることができます(得点は100点に設定)。

●ボーナス倍率(無，有)

これもオリジナルにはないルールです。ボーナス倍率を「有」にすると，カードを2列，3列と同時に揃えたとき，得点も2倍，3倍になります。たとえば，場の中心にカードを置くことで横方向にストレート(10点)，縦方向にフラッシュ(15点)ができた場合，得点は(10+15)×2＝50になるということです。

●手札の向き(表，裏)

手札の向きを設定します。

●カードの動き(無，有)

手札の山からカードを取ってくる際に，マウスカーソルまでカードを少しずつ動かすか一気に移動するかを設定します。

●効果音(無，有)

効果音の有無を設定します。

以上です。

ちなみに，設定画面のときに右クリックした場合は，順位表→その前にプレイした場→設定画面の順(クリックするたびに)画面が変わります。順位表には上位3人の得点とプレイ状態が記録されます。プレイ状態はおまけ機能の上から3つの状態を左から順番に表しています。

〈参考文献〉
トランプ トーレン・ミニブックス刊行会編，トーレン出版部発行

表1 変数表（グローバル変数）

整数型	
sc	得点
bai	得点の倍率
dsc	得点の増分
ct0	ウェイトの長さ
ctl	カードが移動するときのコマ数
mx,my	マウスカーソルの座標
inited	フラグ。FM音源が初期化されていなかったら0，されていたら1
文字型	
t	オールマイティ
配列整数型	
cd( )	場札の任意の一列の内容
pts( )	役の点数
hsc( )	高得点の上位3つ
yknm( )	完成した役の数
tecd( )	手札の内容
bacd(,)	場札の内容
md( )	ゲームの設定状況
	md(1)：ジョーカーの数
	md(2)：ボーナス倍率の有無
	md(3)：手札の向き
	md(4)：カードの動き
	md(5)：効果音の有無
配列char型	
grd( )	ランキングのグラフィックデータ
配列文字型	
st( )	ゲームの設定状態
	st(0)：ハイスコア時の設定
	st(1)：現在の設定
sel(,)	ゲーム設定項目の選択肢

リスト1

```
  10 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
  20 /*                                                /*
  30 /*            ひとりポーカー  +                   /*
  40 /*                      Programmed by K.Furuki    /*
  50 /*                                                /*
  60 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
  70 int sc,bai,dsc,mx,my,inited=0,ct0=1,ct1=16
  80 str t[64]
  90 dim int cd(5),bacd(4,4),tecd(53),yknm(8)
 100 dim int hsc(3)={ 35,20,10 },md(6)={ 0,0,0,0,0,0,0 }
 110 dim int pts(8)={ 100,50,30,20,15,10,6,3,1 }
 120 dim char grd(43199)
 130 dim str st(1)={ "","" }
 140 dim str sel(6,2)={ "",   "始","","",   "0","1","2",
 150           "無","有","",  "裏","表","",  "無","有","",
 160           "無","有","",  "終","",""     }
 170 if 1=1=1 then ct0=2:ct1=80
 180 init()
 190 cls
 200 /*=========================
 210 /*        MAIN
 220 /*=========================
 230 while config()<6
 240   prep()
 250   play()
 260   ranking()
 270 endwhile
 280 mouse(0)
 290 screen 1,1,1,1
 300 print "Thank you for playing!"
 310 end
 320 /*=========================
 330 /*            初期設定
 340 /*=========================
 350 func init()
 360   int i,y
 370   str s0="  0無裏   0  0  0  0",s1="  1  1  1"
 380   dim str t0(6)={"ゲーム開始","JOKERの数","ボーナス倍率",
 390            "手札の向き","カードの動き","効果音","終了" }
 400   dim str t1(1)={ "最高得点"," 得  点 " }
 410   dim str t2(8)={ "FIVE CARD","STRAIGHT FLUSH",
 420            "FOUR CARD","FULL HOUSE","FLUSH","STRAIGHT",
 430            "THREE CARD","TWO PAIR","ONE PAIR" }
 440   screen 1,1,1,1
 450   console ,,0
 460   palet(1,1)
 470   palet(2,&H200)
 480   palet(6,&H600)
 490   randomize(val(mid$(time$,4,2)+right$(time$,2)))
 500   cd(5)=-1
 510   vpage(0)
 520   /*------  設定画面
 530   print 1
 540   apage(0)
 550   bd(152,34,359,477,0)
 560   box(152,34,359,477,1)
 570   bd(192,50,319,97,1)
 580   smbl(200,58,"ひとりポーカー",1,2,13)
 590   for i=0 to 6
 600     y=i*52
 610     smbl(184,110+y,t0(i),1,2,15)
 620     bd(296,102+y,327,149+y,1)
 630     smbl(304,110+y,sel(i,0),1,2,15)
 640   next
 650   /*------  順位表
 660   print 2
 670   apage(1)
 680   home(1,456,364)
 690   t="順位  得点  J倍手  5C SF 4C FH FL ST 3C 2P 1P"
 700   bd(0,0,399,215,0)
 710   box(0,0,399,215,1)
 720   smbl(16,16,t,1,2,9)
 730   line(16,52,383,52,14)
 740   line(17,53,384,53,1)
 750   smbl(24, 56,"1st     35"+s0+"  1  1"+s1,1,2,15)
 760   smbl(24, 92,"2nd     20"+s0+"  0  1"+s1,1,2,15)
 770   smbl(24,128,"3rd     10"+s0+"  0  0"+s1,1,2,15)
 780   line(16,164,383,164,14)
 790   line(17,165,384,165,1)
 800   smbl(24,168,"YOU      0"+s0+string$(5,"  0"),1,2,15)
 810   get(0,0,399,215,grd)
 820   /*------  メイン画面
 830   print 3
 840   apage(2)
 850   fill(0,0,511,511,8)
 860   for i=0 to 1
 870     y=8+i*108
 880     bd(304,y,479,y+99,0)
 890     fill(316,y+24,387,y+39,5-i*2)
 900     smbl(320,y+16,t1(i),1,2,15)
 910     bd(320,y+52,383,y+83,1)
 920   next
 930   sc=35
 940   putcnf(0)
 950   bd(400,132,463,163,1)
 960   fill(304,224,479,399,0)
 970   for i=0 to 8
 980     y=240+i*16
 990     smbl(320,y,t2(i),1,1,15)
1000     t=itoa(pts(i))
1010     smbl(464-len(t)*8,y,t,1,1,15)
1020   next
1030   /*------  一番奥の画面
1040   apage(3)
1050   bd(304,224,479,399,0)
1060   mouse(4)
1070   mouse(1)
1080 endfunc
1090 /*=========================
1100 /*       CONFIG
1110 /*=========================
1120 func config()
1130   int l,r,n
1140   msarea(296,102,327,461)
1150   setmspos(312,126)
1160   apage(0)
1170   vpage(13)
1180   repeat
1190     mswt(1)
1200     repeat
1210       msstat(n,n,l,r)
1220       mspos(mx,my)
1230     until r or (l and ((my-102) mod 52<48))
1240     snd(2)
1250     if r then {
1260       vpage(14):mswt(0)
1270       vpage(12):mswt(0)
1280       vpage(13)
1290     } else {
1300       my=(my-102)¥52
1310       if md(my)=2 then md(my)=0 else md(my)=md(my)+1
1320       if sel(my,md(my))="" then md(my)=0
1330       fill(303,109+my*52,321,143+my*52,4)
1340       smbl(304,110+my*52,sel(my,md(my)),1,2,15)
1350     }
1360   until my=0 or my=6
1370   if md(5) and inited=0 then musinit()
1380   return(my)
1390 endfunc
1400 /*=========================
1410 /*         準備
1420 /*=========================
1430 func prep()
1440   int i,j,a,b,k
1450   vpage(12)
1460   msarea(32,8,287,503)
1470   apage(1)
1480   fill(0,0,399,215,0)
1490   apage(3)
1500   fill(32,8,287,503,8)
1510   sc=0
1520   for i=0 to 8
1530     yknm(i)=0
1540   next
1550   putcnf(1)
1560   fill(32,8,287,503,0)
1570   fill(304,408,479,503,8)
1580   /*------  カードを切る
1590   for i=0 to 51
1600     tecd(i)=i+1
1610   next
1620   tecd(52)=53
1630   tecd(53)=53
1640   for i=0 to 99
1650     a=rnd()*(52+md(1)):b=rnd()*(52+md(1))
1660     k=tecd(a)
1670     tecd(a)=tecd(b)
1680     tecd(b)=k
1690   next
1700   /*------  画面準備
1710   for j=0 to 4
1720     for i=0 to 4
1730       box(32+i*52,8+j*100,79+i*52,103+j*100,15,&H3333)
1740       bacd(i,j)=-1
1750     next
1760   next
1770   for i=0 to 51+md(1)
1780     c_put(432-i*2,408,tecd(i)*md(3))
1790     snd(1)
1800   next
1810 endfunc
1820 /*=========================
1830 /*          PLAY
1840 /*=========================
1850 func play()
1860   int x,y,rest,mycd,l,r,n
1870   rest=51+md(1)
1880   repeat
1890     x=432-rest*2
1900     mycd=tecd(rest)
1910     home(1,513-x,105)
1920     apage(1)
1930     box(0,0,48,96,0)
1940     if md(4) then n=md(3) else n=1
1950     c_put(1,1,mycd*n)
1960     vpage(14)
1970     apage(2)
1980     box(x,408,x+1,503,8)
1990     c_put(x+2,408,tecd(rest-1)*md(3))
2000     if md(4) then {
2010       repeat
2020         x=x-1
2030         home(1,512-x,105)
2040       until x=304
2050       apage(1)
2060       c_put(1,1,mycd)
2070     } else apage(1)
2080     box(0,0,48,96,14)
2090     snd(3)
2100     wait(10)
2110     cdmv()
2120     while 1
2130       repeat
2140         msps()
2150         home(1,537-mx,561-my)
2160         msstat(n,n,l,r)
2170       until l+r
2180       if (mx-32) mod 52>47 then continue
2190       if (my-8) mod 100>95 then continue
2200       x=(mx-32)¥52
2210       y=(my-8)¥100
2220       if bacd(x,y)=-1 then break
2230     endwhile
2240     apage(2)
2250     vpage(12)
2260     c_put(32+x*52,8+y*100,mycd)
2270     bacd(x,y)=mycd-1
2280     rest=rest-1
2290     snd(3)
2300     wait(25)
2310     /*----  横・縦・斜めの列を調べる
2320     dsc=0:bai=1 xor md(2)
2330     check(0,1,y,0,y)
2340     check(x,0,0,1,5+x)
2350     if x=y then check(0,1,0,1,10)
2360     if x+y=4 then check(0,1,4,-1,11)
2370     if md(2)*dsc then sc=sc+dsc*bai:putsc(1)
2380   until rest=26+md(1)
2390 endfunc
2400 /*-----------------  check : カードが5枚あるか調べる
2410 func check(x0,xx,y0,yy,n)
2420   int i
2430   for i=0 to 4
2440     cd(i)=bacd(x0+xx*i,y0+yy*i)
2450     if cd(i)=-1 then break
2460   next
2470   if i=5 then addsc(n)
2480   if md(2)=0 and dsc>0 then {
2490     sc=sc+dsc:putsc(1)
2500     dsc=0
2510   }
2520 endfunc
2530 /*-----------------  cdmv : カードを マウスカーソル に移動
2540 func cdmv()
2550   int i,x,y
2560   if md(4)=0 then {
2570     vpage(12)
2580     msps()
2590     home(1,537-mx,561-my)
2600     wait(1)
```

```
2610     vpage(14)
2620   } else {
2630     for i=1 to ct1-1
2640       msps()
2650       x=(mx-328#)/ct1*i+304
2660       y=(my-456#)/ct1*i+408
2670       home(1,513-x,513-y)
2680     next
2690   }
2700 endfunc
2710 /*----------------  putsc : スコア表示
2720 func putsc(y)
2730   y=60+108*y
2740   apage(2)
2750   fill(400,y,466,y+34,4)
2760   smbl(464-len(itoa(sc))*16,y,itoa(sc),2,2,15)
2770 endfunc
2780 /*----------------  putcnf : ゲーム設定をスコア横に表示
2790 func putcnf(y)
2800   str s
2810   putsc(y)
2820   s=mid$("０１２",md(1)*2+1,2)
2830   s=s+mid$("無有",md(2)*2+1,2)
2840   s=s+mid$("裏表 ",md(3)*2+1,2)
2850   if s=st(y) then return()
2860   st(y)=s
2870   y=68+y*108
2880   fill(328,y,378,y+18,4)
2890   smbl(328,y,s,1,1,15)
2900 endfunc
2910 /*----------------  ranking : 順位表を表示
2920 func ranking()
2930   int i,y
2940   apage(1)
2950   home(1,456,364)
2960   t=right$("    "+itoa(sc),4)+"  "+st(1)+" "
2970   for i=0 to 8
2980     t=t+" "+right$(" "+itoa(yknm(i)),2)
2990   next
3000   put(0,0,399,215,grd)
3010   i=3
3020   while sc>=hsc(i-1)
3030     i=i-1
3040     y=i*36+56
3050     if i<2 then {
3060       hsc(i+1)=hsc(i)
3070       get(64,y,386,y+34,grd)
3080       put(64,y+36,386,y+70,grd)
3090     }
3100     if i=0 then putcnf(0):apage(1):break
3110   endwhile
3120   fill(64,168,386,202,4)
3130   smbl(64,168,t,1,2,15)
3140   if i<3 then {
3150     get(64,168,386,202,grd)
3160     put(64,y,386,y+34,grd)
3170   }
3180   get(0,0,399,215,grd)
3190   if i<3 then {
3200     hsc(i)=sc
3210     smbl(24,y,mid$("1st2nd3rd",i*3+1,3)+"  "+t,1,2,5)
3220   }
3230   vpage(14)
3240   mswt(0)
3250 endfunc
3260 /*========================
3270 /*          役判定
3280 /*========================
3290 func yaku()
3300   int i,j,m,n,yk=1
3310   dim int pair(1)={ 0,0 }
3320   /*------  フラッシュか?
3330   n=0
3340   while cd(n)=52
3350     n=n+1
3360   endwhile
3370   n=cd(n)¥13
3380   for i=0 to 4
3390     if cd(i)<52 and cd(i)¥13<>n then yk=5:break
3400   next
3410   /*------  カードを昇順に並べる
3420   for i=0 to 4
3430     if cd(i)<52 then cd(i)=cd(i) mod 13
3440   next
3450   for j=0 to 3
3460     for i=j+1 to 4
3470       if cd(j)>cd(i) then {
3480         n=cd(j)
3490         cd(j)=cd(i)
3500         cd(i)=n
3510       }
3520     next
3530   next
3540   /*------  ストレートか?
3550   if cd(0)=0 and cd(1)=9 then m=1 else m=0
3560   n=0
3570   for i=m to 3
3580     if cd(i)+1=cd(i+1) or cd(i+1)=52 then continue
3590     if cd(4-n)=52 and cd(i)+2=cd(i+1) then {
3600       n=n+1
3610     } else {
3620       yk=yk+3
3630       break
3640     }
3650   next
3660   /*------  その他
3670   if yk=8 then {
3680     i=0
3690     for j=0 to 1
3700       while cd(i)<>cd(i+1) and i<4
3710         i=i+1
3720       endwhile
3730       while cd(i)=cd(i+1) and cd(i)<52
3740         pair(j)=pair(j)+1
3750         i=i+1
3760       endwhile
3770     next
3780     switch pair(0)*pair(1)
3790       case  2:yk=3:break  /* FULL HOUSE
3800       case  1:yk=7:break  /* 2 PAIR
3810       default:/* others
3820         switch pair(0)+pair(1)
3830           case 3:yk=2:break  /* 4 CARD
3840           case 2:yk=6:break  /* 3 CARD
3850           case 0:yk=9        /* ﾌﾞﾀ
3860         endswitch
3870     endswitch
3880     for i=3 to 4
3890       if cd(i)=52 then {      /* ジョーカーがある場合
3900         switch yk
3910           case 2:yk=0:break  /* 4 CARD → 5 CARD
3920           case 6:yk=2:break  /* 3 CARD → 4 CARD
3930           case 7:yk=3:break  /* 2 PAIR → FULL HOUSE
3940           case 8:yk=6:break  /* 1 PAIR → 3 CARD
3950           case 9:yk=8        /* ﾌﾞﾀ     → 1 PAIR
3960         endswitch
3970       }
3980     next
3990   }
4000   return(yk)
4010 endfunc
4020 /*----------------  addsc : 点数を加算する
4030 func addsc(mode)
4040   int yk,i,j,x,y
4050   yk=yaku()
4060   if yk=9 then return()
4070   dsc=dsc*md(2)+pts(yk)
4080   yknm(yk)=yknm(yk)+1
4090   t=itoa(dsc)
4100   if md(2) then bai=bai+1:t=t+"  "+itoa(bai)
4110   mark(mode,13)
4120   fill(312,240+yk*16,471,255+yk*16,3)
4130   apage(2)
4140   snd(3+bai)
4150   smbl(456-len(t)*8,140,t,1,1,13)
4160   if md(2) then {
4170     for j=0 to 1
4180       y=144-j*2
4190       for i=0 to 1
4200         x=440-i-j*2
4210         line(x,y,x+5,y+11,1+j*12)
4220         line(x+5,y,x,y+11,1+j*12)
4230       next
4240     next
4250   }
4260   wait(45)
4270   mark(mode,5)
4280   apage(2)
4290   fill(402,134,461,161,4)
4300   apage(3)
4310   fill(312,240,471,383,4)
4320 endfunc
4330 /*----------------  mark : 役のできた列をマークする
4340 func mark(mode,col)
4350   int a,b,i
4360   apage(3)
4370   if mode<5 then {
4380     /*----  横
4390     a=mode*100
4400     fill(80,48+a,239,63+a,col)
4410   } else if mode<10 then {
4420     /*----  縦
4430     a=(mode-5)*52
4440     fill(48+a,104,63+a,407,col)
4450   } else {
4460     /*----  斜め
4470     if mode=10 then a=56:b=263 else a=263:b=56
4480     for i=0 to 15
4490       line(a,48+i,b,447+i,col)
4500     next
4510   }
4520 endfunc
4530 /*========================
4540 /*          その他
4550 /*========================
4560 /*----------------  bd : 掲示板の表示
4570 func bd(x0,y0,x1,y1,mode)
4580   int a,c0,c1
4590   if mode then a=2:c0=6:c1=2 else a=8:c0=2:c1=6
4600   fill(x0,y0,x1,y1,c0)
4610   fill(x0+a,y0+a,x1-a,y1-a,4)
4620   a=a-1
4630   line(x0,y1,x0+a,y1-a,c1)
4640   line(x1,y0,x1-a,y0+a,c1)
4650   paint(x0,y0,c1)
4660 endfunc
4670 /*----------------  msps : ﾏｳｽｶｰｿﾙ の位置を返す（補正付）
4680 func msps()
4690   mspos(mx,my)
4700   if mx<56 then mx=56 else if mx>264 then mx=264
4710   if my<56 then my=56 else if my>456 then my=456
4720 endfunc
4730 /*----------------  mswt : マウスボタンによる待ち
4740 func mswt(a)
4750   int l,r,n
4760   if a=0 then mouse(2):mswt(1)
4770   repeat
4780     msstat(n,n,l,r)
4790   until l or r xor -a
4803   if a=0 then snd(2):mouse(1)
4810 endfunc
4820 /*----------------  musinit : トラック初期設定
4830 func musinit()
4840   int i
4850   str t0
4860   m_init()
4870   for i=1 to 7
4880     m_alloc(i,99)
4890     m_assign(i,i)
4900   next
4910   m_trk(1,"@23o4l64v13g.a.")
4920   m_trk(2,"@15o4l32v12a")
4930   m_trk(3,"@59o4l32v15f.")
4940   t0="@39o3l48v13"
4950   t="c&d&e&f&g&a&b&<":t=t+t+t
4960   for i=0 to 3
4970     m_trk(i+4,t0+mid$(t,i*2+1,28+i¥i)+"24")
4980   next
4990   inited=1
5000 endfunc
5010 /*----------------  smbl : 文字に影を付けて表示
5020 func smbl(x,y,t;str,h,v,col)
5030   int a,b
5040   for b=2 to 3
5050     for a=2 to 3
5060       symbol(x+a,y+b,t,h,v,1,1,0)
5070     next
5080   next
5090   for b=0 to 1
5100     for a=0 to 1
5110       symbol(x+a,y+b,t,h,v,1,col,0)
5120     next
5130   next
5140 endfunc
5150 /*----------------  snd : 効果音を鳴らす
5160 func snd(a)
5170   if md(5) and inited then m_play(a)
5180 endfunc
5190 /*----------------  wait : 無駄ループ
5200 func wait(t)
5210   int i
5220   for i=0 to t*99*ct0:next
5230 endfunc
```

BACK ISSUES バックナンバー案内

ここには1993年9月号から1994年8月号までをご紹介しました。現在1993年9～12月号，1994年1，3～8月号の在庫がございます。バックナンバーはお近くの書店にご注文ください。定期購読の申し込み方法は144ページを参照してください。

1993

9月号
特集　光学式磁気円盤MO

連載
DōGA CGアニメーション講座/こちらシステムX探偵事務所
響子 in CGわ～るど/ショートプロ/大人のためのX68000
ハードエ作/Computer Music入門/ANOTHER CG WORLD

●新製品紹介　OS-9/X68030
LIVE in '93　ファイナルファンタジーVのテーマ/銀河鉄道999/アルスラーン戦記IIより　汗血公路/ちょうちょ
THE SOFTOUCH　悪魔城ドラキュラ/コットン/ダーク・オデッセイ 他
全機種共通システム　7並べ/SLANG再々掲載

10月号
特別企画　秋祭りPRO-68K

連載
ハードコア3D/Computer Music入門/マシン語プログラミング
DōGA CGアニメーション講座/こちらシステムX探偵事務所
響子 in CGわ～るど/ショートプロ/吾輩はX68000である

●特別付録　秋祭りPRO-68K（5"2HD）
●SCSIバックンTOWER JACK
LIVE in '93　未来予想図II/OutRunより　PASSING BREEZE
THE SOFTOUCH　コットン/The World of X68000/あにまーじゃんV3
全機種共通システム　シューティングゲームコアシステム作成法(4)

11月号
特集　ポリゴナイザSLASHの活用

連載
ハードコア3D/Computer Music入門/ファイル共有の実験と実践
こちらシステムX探偵事務所/目指せジョイスティックの星
響子 in CGわ～るど/ショートプロ/大人のためのX68000

●新製品紹介　Easydraw SX-68K
OS-9 Ultra C/Technical Tool Kit
LIVE in '93　渚のアデリーヌ/エロティカ・セブン
THE SOFTOUCH　ぶたさん/ダイアット・ヴァークス
全機種共通システム　S-OSで学ぶZ80マシン語講座(1)

12月号
特集　古今東西ゲーム議論

連載
ハードコア3D/マシン語プログラミング/響子 in CGわ～るど
DōGA CGアニメーション講座/こちらシステムX探偵事務所
ショートプロ/Computer Music入門/ファイル共有の実験と実践

●新製品紹介　MATIER ver.2.0
C Compiler PRO-68K ver.2.1 NEW KIT
LIVE in '93　クリスマス・イブ/星に願いを
THE SOFTOUCH　ネメシス'90改/項劉記/スーパーリアル麻雀PII&PIII
全機種共通システム　エディクアセンブラREDA再掲載

1994

1月号
特集　Z-MUSICシステムver.2.0

連載
ハードコア3D/ゲーム作りのKNOW HOW/響子 in CGわ～るど
DōGA CGアニメーション講座/こちらシステムX探偵事務所
ショートプロ/Computer Music入門/ファイル共有の実験と実践

●特別企画　ANOTHER CG WORLD in Hong Kong
LIVE in '94　LAST WAVE/スターウォーズ/明日への扉/夢路より 他
THE SOFTOUCH　ストリートファイターIIダッシュ/餓狼伝説2/ドラゴンバスター/X68000傑作ゲーム選
全機種共通システム　S-OSで学ぶZ80マシン語講座(2)

2月号（品切れ）
特集　X-BASICとグラフィック

連載
ハードコア3D/ワンチップIC/響子 in CGわ～るど
DōGA CGアニメーション講座/こちらシステムX探偵事務所
ショートプロ/Computer Music入門/ANOTHER CG WORLD

●新製品紹介　ハイパーピクセルワークス
LIVE in '94　ランス3/新宿駅，巣鴨駅の発車メロディ/ピコー・ソング
THE SOFTOUCH　キーパー/マッドストーカーX68/餓狼伝説2 他
全機種共通システム　S-OSで学ぶZ80マシン語講座(3)
YGCSver.0.20リファレンスマニュアル

3月号
特別企画　ひなまつりPRO-68K

連載
ハードコア3D/マシン語プログラミング/ゲーム作りのKNOW HOW
DōGA CGアニメーション講座/こちらシステムX探偵事務所
ショートプロ/響子 in CGわ～るど/ファイル共有の実験と実践

●特別付録　ひなまつりPRO-68K（5"2HD）
●新製品紹介　ビデオPC for X680x0
LIVE in '94　THEME FROM WINNING RUN/スターフォースアレンジ版
THE SOFTOUCH　卒業/マッドストーカーX68/B-FIELD! 他
全機種共通システム　S-OSで学ぶZ80マシン語講座(4)

4月号
特集　SX-WINDOWの活用

連載
ハードコア3D/こちらシステムX探偵事務所
DōGA CGアニメーション講座/響子 in CGわ～るど
ショートプロ/ローテク工作/ANOTHER CG WORLD

●決定！　1993年度GAME OF THE YEAR
●新製品紹介　ビデオ入力ユニットCZ-6VS1
LIVE in '94　宇宙戦艦ヤマト/プロジェクトA子
THE SOFTOUCH　ジオグラフシール/ぷよぷよ/レッスルエンジェルス2　他
全機種共通システム　S-OSで学ぶZ80マシン語講座(5)

5月号
特別企画　こいのぼりPRO-68K
第9回言わせてくれなくちゃだワ

連載
ハードコア3D/響子 in CGわ～るど/ショートプロ
DōGA CGアニメーション講座/ファイル共有の実験と実践
こちらシステムX探偵事務所/ANOTHER CG WORLD

●特別付録　こいのぼりPRO-68K（5"2HD）
●新製品紹介　WorkroomSX-68K/開発キットツール集
LIVE in '94　ロード/時間旅行
THE SOFTOUCH　大魔界村/アルゴスの戦士/ジオグラフシール　他

6月号
特集　X68000と仲間たち

連載
ハードコア3D/響子 in CGわ～るど/ショートプロ
ローテク工作/ファイル共有の実験と実践
こちらシステムX探偵事務所/ANOTHER CG WORLD

●第5回Oh!Xアンケート分析大会
●新製品紹介　F-Calc for x68k
LIVE in '94　キャミイのテーマ/The End of Love
THE SOFTOUCH　スーパーリアル麻雀PIV/あすか120% BURNING Fest.他
全機種共通システム　YGCS ver.0.30

7月号
特集　入門コンピュータミュージック

連載
響子 in CGわ～るど/ショートプロ/ゲーム作りのKNOW HOW
ローテク工作/システムX探偵事務所/マシン語プログラミング
DōGA CGアニメーション講座/ファイル共有の実験と実践

●特別付録　CGA入門キット「GENIE」
●実用講座　Photo CDでカードを作る
LIVE in '94　宇宙刑事ギャバン/究極戦隊ダダンダーン/スティング　他
THE SOFTOUCH　麻雀航海記/雀神クエスト/The World of X68000 II 他
全機種共通システム　シューティングゲーム作成講座(1)

8月号
特集　Graphic Movement

連載
響子 in CGわ～るど/ショートプロ/ハードコア3D
ローテク工作/ANOTHER CG WORLD/善バビ
DōGA CGアニメーション講座/石の言葉，言葉の夢

●新製品紹介　X-SIMM VI/Mu-1 GS
SX-WINDOW ver.3.1
LIVE in '94　PURE GREEN/Ridge racer(POWER REMIX)
THE SOFTOUCH　Mr.Do!/Mr.Do! vs UNICORNS/レッスルエンジェルス3
全機種共通システム　シューティングゲーム作成講座(2)

新製品紹介

X68030 D'ash

Kioi Makoto 紀尾井 誠

X68030を33MHzにクロックアップ。満開製作所謹製の高速マシン，その名もX68030 D'ash。限定50台のチューンドマシンです。しかし，やっぱり「turboほど速くはない」という意味なんでしょうかね？

最近はアクセラレータの話題が多くなってきた。以前は物好きな人がクロックアップしている程度だったのに，ここへきてなぜ本体改造関係の話題まで出ているのだろうか。

基本的な事実として，X68030はX68000シリーズの最終形態として生み出されたマシンだということがある。

正式な話ではないにせよ，Xシリーズの総帥たる鳥居氏もX68000シリーズはX68030で終わりという談話をしている。だから，いくら待っててもX68040とかX68060は現れない。現状のアーキテクチャのままでCPUだけ高速にしても画期的な高性能にはならない。昔のAT互換機を見れば状況はわかるだろう。

で，X68000の10MHz機をお使いの方には申し訳ないが，X68030でもまだ遅いと思っている人が世の中にはいるものである。

X68000用のソフトならX68030で十分高速に動作させることができる。しかし，X68030でもちょっと重めの処理というのはあるし，SX-WINDOW環境を本格的に使うと心許ないというのも事実である。

まだX68030を極めたソフトというのは現れていないが，ソフト開発が全体的にX68000との互換路線を重視している以上，あまりX68030独自にソフトを作るわけにもいかない。今度ばかりはソフトでなんとかするというわけにもいかないだろう。かといって純正のアクセラレータなどは出そうにもないし……というのが理由のひとつだ。

もうひとつの理由はX68030自体が結構速いという事実にある。そもそもチューンアップというのは「遅いから速くする」人よりも「速いものをもっと速くする」人のほうが圧倒的に多いものだ。

さらに，次期X680x0が現れないということはわかっても，じゃあ新しい次世代マシンはいつ出るのかというと，これもはなはだ不透明な話である。はっきりいえるのは，もし来月出るとわかっていたら，いま無理にいじる人はいないだろうということだけだ。

外見は同じだけど……(製品版にはシール付属)

で，こういった背景から生まれたのが040turboやHARP/FXなどであり，今回紹介する満開製作所のX68030 D'ashもそういったもののひとつである。

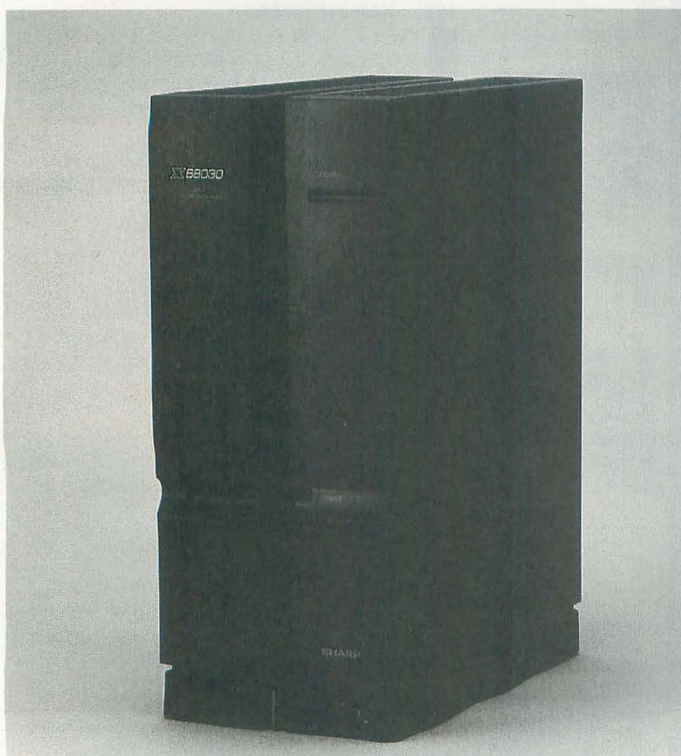
X68030-33MHz改造

D'ashの仕様を簡単に説明しよう。

CPUクロックを33MHzに上げている。同時にCPUも従来の68EC030 (25MHz) から68030 (33MHz) に換装されている。さらに33MHz版の数値演算コプロセッサ68882も最初から装着されている。そのほか，メモリの動作モードが変更され，一部のマシンではいわゆる「030ぴーんち」回避の処置が行われているが，それ以外の部分は無改造だ。25MHzとの切り替えスイッチなども用意されていない。そのぶん電気的な安定度は高くなっている。ちなみに，外観はまったく同じだ。

X68030高速化でひとつの限界と目されているのが37MHz，それ以上となるとオシレータを取り換えるだけではすまなくなる。安定動作を求めるなら35MHz以下，CPUをオーバードライブせずコストを重視すれば33MHzに落ち着く。

なお，CPUクロック以外にシステムクロックを上げて性能を向上させることも一部では行われている。しかし，システムクロックを変えると周辺機器にまで影響を及ぼす可能性がある。特にウエイトポートとして使用されるジョイスティックの読み出し速度が変わると，CPU速度に依存しないアプリケーションの作成が困難になるのであまりおすすめはできない。

こうして見ると，単に高性能を目指しているのではなく，安定動作に重点をおいていることがわかる。コストとリスクとパフォーマンスを考えれば結構バランスの取れた設定といえるだろう。仕様として特に無理なところはない。むしろ，本来X68030は

これくらいの仕様の製品であってもよかったのではないかという気すらしてくる。

ハード工作にある程度慣れた人なら，同じ仕様のものを作ることはさして困難ではない。33MHz版の68030と68882を買ってきて(5，6万円くらいか？)，オシレータを換えて，ちょこちょこいじって，運がよければ動く。

慣れてない人にはちょっとばかり困難である。まあ，CPUやFPUの入れ方を間違えない限り致命的な故障にはならないとは思うが，最悪の場合でなくとも保証の対象外であることは間違いない。

X68030 D'ashとはいっても，ハード的に特殊な仕様を持たせたマシンではない。特殊なのはそれの動作を保証しているということだ。トラブルがあれば純製品と同等(以上？）のサポートが受けられる。

もちろん動作保証などはされていないが，この上に040turboを載せることも可能だ。68040は33MHz版に換えておいたほうがよいのはいうまでもない(といいつつ25MHz版のまま使っているが，なんともない。ただし冷却は徹底的にね)。

ベンチマークテスト

さっそく性能評価をやってみよう。今回使用したのはスタンフォードベンチマークテストだ(XC ver.2.1でコンパイルしたもの)。図1にいろいろまとめてみる。数値演算ドライバはFLOAT2による結果である。X68030でキャッシュを使用しない状況というのはほぼないと思われる。よってすべてキャッシュON時の結果である。

040turboでもキャッシュなしでの使用はほぼ考えられない。問題のあるプログラムがあったとしたらパッチなどで対処するか，そのツールを使わなくなるだけだろう。ということで，実使用上，コピーバックモードを常用してもあまり支障がない。一応，ライトスルーモードとコピーバックモード両方の値を掲載しておいた。石上版アクセラレータの値は暫定的なものである。これはデータキャッシュは動くようになったが，メモリウエイトはそのまま残されているバージョンのテスト基板のものだ。

図の見方を説明しよう。だいたいにおいてintよりfloatのほうが伸びが鈍い。これを見て，

「実数演算って速くなんないんだね」

「そうだね」

という会話はしないように。同じコードを実行させるためにすべてFLOAT2で測定したものである。

ではどういう風に見るかというと，まずintは通常のプログラムの実行速度を示している。XCの出したオブジェクトというのもミソだ。それに対してfloatはFLOAT2をめいっぱい使った処理と考えていい。ご存じのようにFLOAT2のver. 2といえば，68000用にギチギチに最適化されたプログラムである（もっと速いフリーウェアもあるが，たいていはエラーチェックを抜いて速度を稼いでいる)。だから，intとfloatの差はコードの質による高速化の割合だと思っていいだろう。

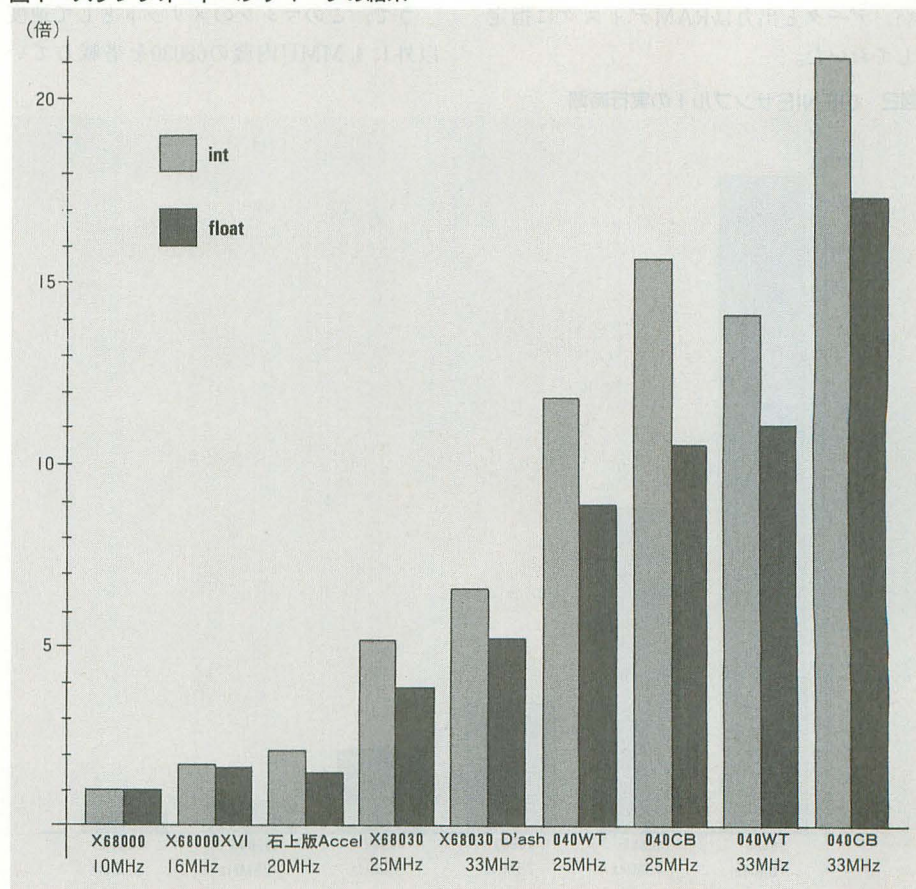
速度的におおまかな目安をつけると，

X68000	1
X68000XVI	2
REDZONE	3
X68030	5
X68030D'ash	7
040turbo	10

のようになるだろうか。

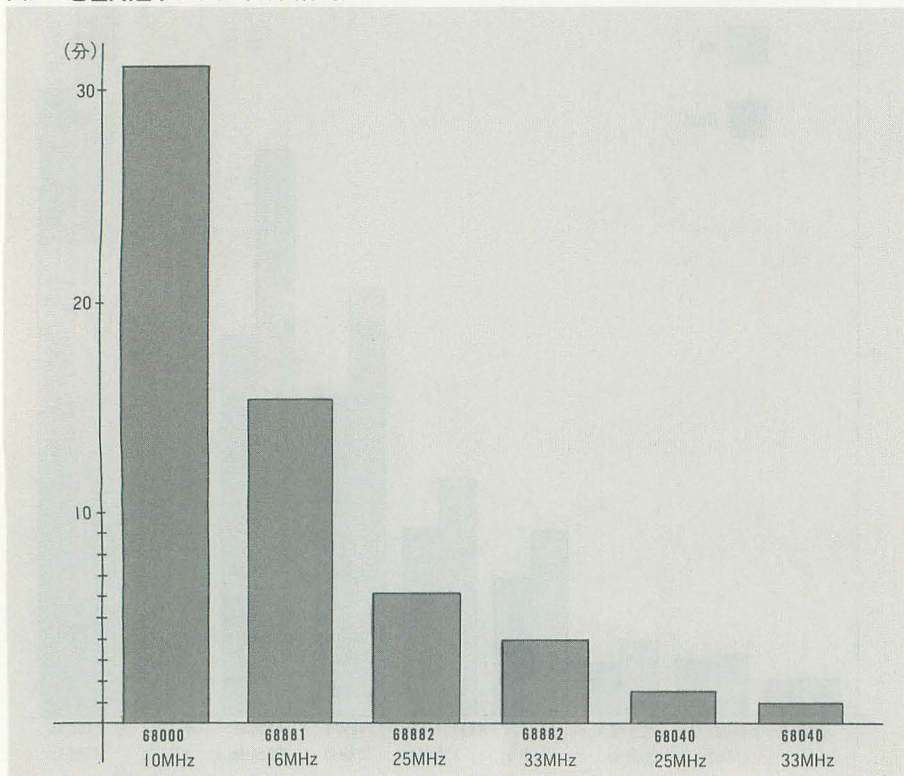
しかし，こういったベンチマークテストと体感速度ではずいぶん開きがあることがある。速くない機種を使っていても遅さはわからないのと同様，遅くない機種を使っていても速さはわからないものである。特にX68030以上の速さは体感では表れにくい。

X68030から移行するとなると，X68000からXVIに移ったときのような劇的な環境変化は感じられないだろう（相当重い処理

図1 スタンフォードベンチマークの結果

をやらないかぎりは)。たとえばX68030ユーザーにD'ashと040turboの2つを(同時にではなく)触らせても,区別できる人は少ないだろう。

そのほか,コプロセッサによる浮動小数点演算の高速化も見逃せない。ここではDōGA CGAシステムのパフォーマンスということで,7月号付録ディスクに収録されたGENIEのアニメーションサンプル1を「すべてあり」のモードで実行したときの時間を計ってみた。X68000シリーズで実数演算を極端に多用する機会というのはほかではあまりないと思われるので(Easy drawなどはかなり速くなるが),もっとも現実的なテストだと思われる。

計測機種はX68000の10MHzとX68000 XVI+68881,X68030+68882,X68030 D'ash,040turbo,D'ash+040turboという構成になっている。キャッシュやレンダラはそれぞれで最適なものを使用している。すべて同じハードディスク上のシステムを使い,データと出力はRAMディスクに指定しておいた。

図2 GENIEサンプル1の実行時間

最大1:31の速度比となっているのがわかるだろう。要するに,

「丸1カ月かかったレンダリングが1日でできる」

ということになる。うむうむ。

ちなみに,これにはディザリングなどの浮動小数点数を必要としない重い処理も含まれているので(先ほどの図から通常処理の速度比は最大1:20程度),実質的な浮動小数点演算の速度差はもっと大きなものになると見ていい。

次にグラフィックRAMのアクセス速度を見てみよう。図3はX-BASICで全画面FILLを100回行ったときの時間(秒)である。インタプリタとはいえ100回のループくらいは無視できる時間だが,かなり荒っぽいテストなので1秒程度の誤差はありうる。まあだいたいのところはわかるだろう。

見てわかるようにたいして速くならない。68040のほうが若干遅めになる。このへんがI/Oコンパチの辛いところだ。

さて,このマシンのメリットとして速度以外にもMMU内蔵の68030を搭載しているという点がある。Human68kは仮想記憶に対応していないので通常使用時にはなんのメリットにもならないが,NetBSD(要するにUNIX)を動かしたいという場合には手軽でよいかもしれない。これは68EC030では動作しないので,使いたかったら本体を開けて,シールドを外して,CPUを引っこ抜いて,買ってきた68030に差し替える必要がある。速度的なメリットもあわせて考えればさらに向いているといえるだろう。

これ以外では本格的にMMUを使ったプログラムというのはまだ見られないようだ。

問題点は?

高速化といってもメリットばかりではない。念のためにちょっとアラ探しをしてみよう。

まずは発熱の問題。高クロックだと,どうしてもICの発熱量が大きくなるのだが,CPU自体については,プラスチックパッケージの68EC030を25MHzで動かしていたときよりセラミックパッケージの68RC030を33MHzで動かしているときのほうがCPUの蓄熱は少ない(放熱がうまくいっている)。問題はCPU以外のカスタムチップなどだ。どうしようもない問題なのだが,特に過熱するものはないようだ。

クロックアップにともない,メモリの動作モードが変更されている。X68030はDRAMのスタティックカラムモードでせこせこ速度を稼いでいる部分があるのだが(数%速くなる),そういった部分は使えなくなる。まあ,誤差の範囲だ。

なお,今回のマシンには25MHzモードは設けられていない。X68030との速度的な互換性を気にする必要はないのだが,そこから派生してXF3を押しながら起動しても10MHz相当のウエイトでは起動できないという症状が起きる(もともと速度の互換性は怪しかったのではあるが)。場合によると速すぎて支障があるソフトも出てくるかもしれない。

速度以外の互換性については,まったく問題ない。ノーマルのX68030で動いてこの

マシンで動かないソフトというのは，よほど変なことをしない限りありえないだろう。

古いゲームを除けば，X68030で支障のあるソフトなどはごく少数にすぎない。なかには困っている人もいるようだが，大半の人にはなんの支障もないのが現状だ。X68030より速くなって新たに支障が出るソフトはさらに少ないだろう。となればX68030互換モードがなくなったところで不都合はあるまい。

そのほか，VCCIの規準はクリアできているのかとかいい出すとキリがないのだが，基板はきっちりシールド版にくるまれているから電磁波の問題は少ないだろう（040turboなどではシールドは外さなければならない）。

誰のためのシステムか

さて，ではどのような人がこの製品を使うべきだろうか。

すでにX68030を持っている人が無理に買い替える必要はない。現状でX68030ユーザーがパワーアップを求めるなら040turboに走るべきだろう。040なら500Kバイト近いZ-MUSICのソースも3.5秒でアセンブルできるし（X68030で12秒），Easydrawを多用する人には特におすすめだ。

ということで，X68000/X68000XVIユーザーが思い切って乗り換えるというセンだろう。パワーアップにはほかにもいくつかの選択肢があるのだが，決め手になるのはやはりメーカー保証ということだろう。「クロックを上げれば速くなる」なんてのは当然の話で，多少のリスクを覚悟すればクロックアップにだいたい比例した成果が得られる。もちろん潜在的な安定度は反比例して下がっていくが。

加えて，安くあげればそれだけリスクも増えてくる。「安全」という，普通は金を出しても買えないものが買えるのなら，多少の出費は惜しむべきではないだろう。

「金で解決できるときはリスクを冒すな」というのが鉄則である。自前でメモリボードを作ったらデータが化け化けとか，クロックアップして快適に使っていたらある日突然動かなくなったとか，トラブルの種は尽きない。当然のことだが25MHz用に設計されているものをハイクロックで動作させたときに，25MHz時と同じ安定度を求めるのは無理な話である。

X68030が結構頑丈に設計されているとかいっても，高速化ではどうしても製品の個体差が出てくるようである。33MHz化は安定動作の見込める選択だが，巷では「結構安全」のうちに入るXVI24MHz化を行ったREDZONEでも実はかなりたくさんのトラブルがあったそうだ。それらはメーカー保証ですべて対処してきたとのことだ。メイン基板交換クラスのトラブルでも保証が効くというのは素晴らしいことだ（ちょっとメーカー側には同情するが……）。あっ，と念のために書いておくけど，040turboは保証の対象外だから，つけるときはあくまでも各自の責任でやるように。

とにかく，保証が効くというのはもっとも安全で堅実な選択肢であろう。ちょっと危ない性能追求時のリスクとメリットを数値化したとき，

10MHz機改造	10	5
XVI24MHz化	3	5
X68030改造	4	3
040turbo	2	8
高解像化	6	10

くらいになるとすれば（ただし高解像化のメリットはSX-WINDOWユーザーに限る），この製品などはリスク0のメリット3といったところになるだろうか。

いくら速くても車検に通らないクルマを買う馬鹿はいないのだ。

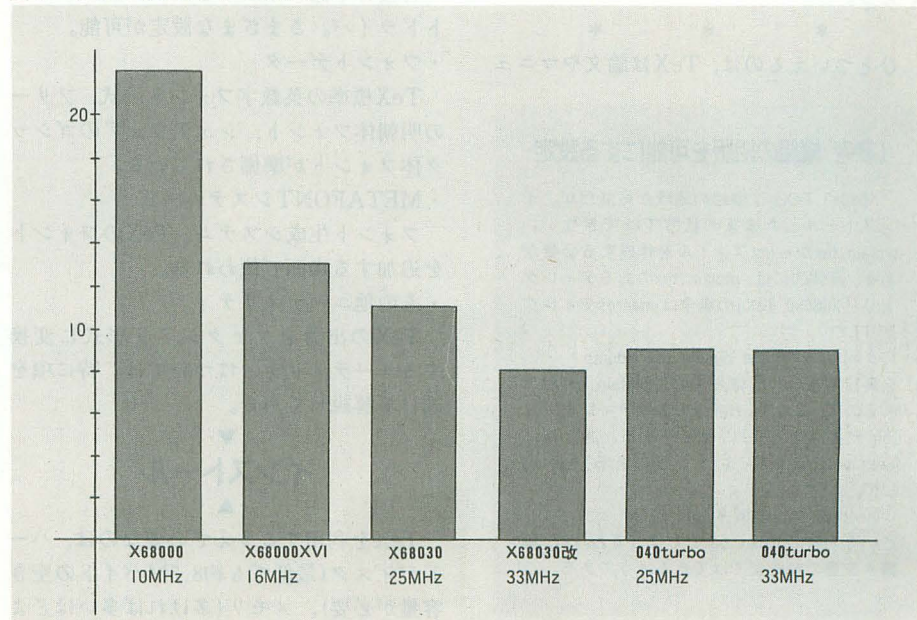
＊　＊　＊

製品と直接は関係ないが，今回のテストで痛切に感じたことは，メモリ4Mバイトではかなりせせこましい使い方しかできないということである。この製品に限らずX68030を選ぶような人は残り8Mバイトのメモリを忘れずに追加してほしい。せっかくメモリが安いのだから迷わず12Mバイトだ。私はそんなに派手なものはシステムに組み込まないほうなのだが，普段使っているシステムが使用できないというのはちょっとショックだった（単にRAMディスクのせいだが）。特にSX-WINDOWは起動するにも四苦八苦だ。まあ，常時50タスクくらい走らせている私も悪いのだが。

結論，やっぱりハイエンドマシンを買うならメモリはフル実装だ。

X68030 D'ash　　368,000円(税別)
満開製作所　　☎0120-887780

図3　グラフィックアクセス時間

NEW BOOKS
新刊紹介

X68k Programming Series(#3)

X680x0 TeX

Tan Akihiko 丹 明彦

世界的に使われている文書作成システムである「TeX」(読み方は「てふ」または「てっく」が一般的)のX68000/030版が発売された。TeXはパブリックドメインなので流通ルートに乗せた実費配布という表現のほうが正確かもしれない。

TeXとは?

さて，そのTeXであるが，どんなものかと問われても答えるのはなかなか難しい。

TeXは文書作成システムではあるが，ワープロではない。TeXは組版システムである。ユーザーの書いた文章を整然とレイアウトし，美しく印刷する。TeX自体が組版の知識を組み込んでいるので，ユーザーがこと細かに指定せずともTeXは文書を美しく出力することができるのである。

TeXの使用感覚はコンパイラに近い。ワープロが画面に映し出されたそのままをプリンタに印刷するのとは対照的に，TeXは原稿(ソース)をテキストエディタで書き，コンパイルすることによって文書を得る。どのように印刷されるかは，プレビューまたは印刷するまではわからない。そういう意味では，TeXはやや玄人向けであるともいえる。

＊　＊　＊

ひとついえるのは，TeXは論文やマニュアルの作成といった場面で圧倒的な人気を博しているという事実である。TeXは論理構造がしっかりした文書，特に大規模な文書を大得意としている。「X680x0 TeX」のマニュアル自体もTeXで書かれている。

製品構成

「X680x0 TeX」は，2冊のマニュアル(ユーザーズガイドとリファレンス)および8枚組のフロッピーディスクからなる。その内訳は，

・インストーラ

全自動のインストールプログラム。

・TeXシステム一式

TeXソースを組版してdvi(DeVice Independent: デバイス非依存)ファイルを生成するプログラム。デバイス非依存とは，文書の情報が画面やプリンタといったデバイスによらない要素だけで構成されているということ。

・デバイスドライバ

dviファイルを解釈して画面やプリンタなどのデバイスに出力する。画面に出力するプログラムはプレビューアと呼ばれる。

・フォントマネージャ

各種日本語フォントを柔軟に扱うフォントドライバ。さまざまな設定が可能。

・フォントデータ

TeX標準の英数字フォント一式，フリーの明朝体フォント，シェアウェアのゴシック体フォントが準備されている。

・METAFONTシステム一式

フォント生成システム。TeXのフォントを追加する場面で使われる。

・その他ユーティリティ

TeXの出力をファクシミリ形式に変換するユーティリティについては，特に項を設けて解説してある。

インストール

TeXを利用するうえで必要なのは，ハードディスク(最低でも約8.5Mバイトの空き容量が必要)，メモリ(多ければ多いほどよい)，プリンタ(解像度がある程度高いもの)，それと入手できるなら市販の日本語フォントである。

「X680x0 TeX」にはインストーラが備えられているので，インストーラを起動していくつかの設問に答えればインストールが始まる。ときどきフロッピーディスクを入れ替えるよう指示されるほかは全自動である。データは圧縮されているので展開作業には1時間単位の時間を要するが，かつてかなりの手間ひまをかけてインストールしたことを思えば楽なものである。

マニュアルは，インストール後にユーザーがTeXに速やかに馴染めるようにガイドや練習メニューにページを割いており，初心者でも安心してTeXの世界に入っていくことができるだろう。

日本語フォント

「X680x0 TeX」には，実用に耐える日本語フォントが付属する。明朝体はフリーで利用できる。ゴシック体はシェアウェアなので，引き続き使う場合はシェアウェア・フィーとして1,500円を作者に支払う必要がある。

そのほか，市販のフォントとして，X68000/030ユーザーにはお馴染みの書体である倶楽部フォント，JGフォントが利用できる。また，アスキーのPC-98シリーズ用「パーソナル日本語TeX」の日本語フォントを利用することもできる。

ここでは，「X680x0 TeX」付属の日本語フォント，および私の手持ちのフォントであるZ'sSTAFF付属フォントとJGフォントを用いた印字サンプルを掲げておく。付属の日本語フォントは印字品質においてそれほど見劣りしていない。

縦書き文書について

pTeX(publishing TeX)は通常の横組だけでなく縦組の文書も作成可能なTeXである。TeXが日本語化されたときから根強くあった要望が実現された格好だ。実際，

(参考)縦組の組版を可能にする設定

「X680x0 TeX」は縦組の組版が可能だが，インストールしたままの状態ではできない。plplain.texからfmtファイルを作成する必要がある。具体的には，plplain.texのあるディレクトリ(「X680x0 TeX」の場合はjmacrosディレクトリ)で，

```
initex.x ¥input plplain.tex ¥dump
```

を実行すると，しばらくしてplplain.fmtができるので，これをvirtex.xのあるディレクトリ(binディレクトリ)にコピーする。あとは，latex.bat(batchディレクトリにある)を改造して，

```
virtex "&plplain ¥input %1"
```

という内容のplatex.batでも作っておけば，縦書き文書の組版まではできるようである。

(参考)キヤノンBJ-220JC用コンフィグレーションファイル

このバブルジェットプリンタは，BJ-10vの上位互換となってはいるが，BJ-10v用のコンフィグレーションファイルを使っていると頻繁に印刷が崩れる。そこでとりあえず，印刷できればいいという感じで大急ぎでコンフィグレーションファイルをでっちあげてみた。何かの役に立てば幸いである。

```
-remark=CANON BJ-220JC PRINT.CFG
-dpi=360
-remark=-TRAM
-remark=-GRAM
-width=2880
-remark=-width=2816
-height=3960
-remark=-height=4032
-remark=-xOffset=-90
-yOffset=-180
-pinBytes=8
-MSBisUpper
-init=¥e@¥e3¥x20¥er%0
-remark=-init=¥e@¥e3¥x20¥er%0
-CRLF=¥n
-remark=-CRLF=¥r¥n
-FF=¥f
-graphic=¥e|B¥x40%2I
-start=
-relative=¥e¥¥%2/2i
-remark=-start=¥e$%2/6i
-remark=-relative=
-repeat=
-dump=LPT
```

縦書きを実現したTeXの処理系はいくつもある。

「X680x0 TeX」はそのpTeXをベースにしており，縦組の組版をする機能は備えている。縦組のマクロも入っている。しかし，現時点ではデバイスドライバが日本語文字の90度回転に対応していないため，出力は不可能であるとマニュアルに記述がある。

試しにやってみると，確かにTeXは通ってdviファイルは生成されるが，プレビューの段階で失敗するようである。

どうしても縦組の印刷をやりたいのであれば，「X680x0 TeX」標準添付以外のデバイスドライバを探すか，他機種でやるしか現時点では方法がないようだ。

マニュアル

マニュアルは「X680x0 TeX」の価値ある部分のひとつである。

初心者向けのインストールやチュートリアルだけでなく，上級者向けのカスタマイズ方法の詳しい解説にもページが割かれている。手持ちのプリンタに対応するコンフィグレーションファイルがなかった場合に対処することも可能になっている。

また，世に知られているTeX関連システムを紹介する章が特に設けられている。楽譜や化学式を扱えるTeXのバリエーションやデバイスドライバ，サポートプログラムといった「TeXファミリー」の概要と入手先など，役立つ情報が記されている。

まとめ

これまでは，TeXは自由に配布できるため，逆にその手の情報にアクセスできる人しか入手できなかった。NIFTY-Serveで関連ファイルを収集して苦労しながらインストールするか，すでに使っている知り合いから分けてもらうくらいしか入手方法がなかった。

本誌宛の読者ハガキのなかにも「TeXって何ですか」とか「TeXを配布してほしい」というものを散見していたが，いまや書店に行っていくばくかの費用を払えば誰にでも入手できる環境が整ったわけである。

また，すでにTeXを使っている人にとっても，きちんと整備されたマニュアルの価値は高いと思われる。このあたりの事情は同シリーズの「#1 X68000 develop.」や「#2 X680x0 libc」と同様である。

「X680x0 TeX」は，TeX初心者から上級TeX使いまで，幅広いユーザーにとって利用価値が高いパッケージといえるだろう。

X68k Programming Series(#3)
X680x0 TeX　9,800円(税込)
ソフトバンク　☎03(5642)8101

3種類のフォントによる印字サンプル

(原寸，使用プリンタはキヤノンBJ-220JCII)

1　見果てぬ夢、TEX Graphics

ご存じのように、TEXは本来文章を美しく組版するためのプログラムであり、描画の機能は極めて限定されています。しかし、その素晴らしい出力を目にして、図や絵を取り込めたらいいなぁと思うひとも多いらしく、LATEX

X680x0 TeXに付属の日本語フォント

1　見果てぬ夢、TEX Graphics

ご存じのように、TEXは本来文章を美しく組版するためのプログラムであり、描画の機能は極めて限定されています。しかし、その素晴らしい出力を目にして、図や絵を取り込めたらいいなぁと思うひとも多いらしく、LATEX

Z'sSTAFFに付属のフォント

1　見果てぬ夢、TEX Graphics

ご存じのように、TEXは本来文章を美しく組版するためのプログラムであり、描画の機能は極めて限定されています。しかし、その素晴らしい出力を目にして、図や絵を取り込めたらいいなぁと思うひとも多いらしく、LATEX

JGフォント

ハードコア3Dエクスタシー(第11回)

SIDE A

動きのある車のための表示系

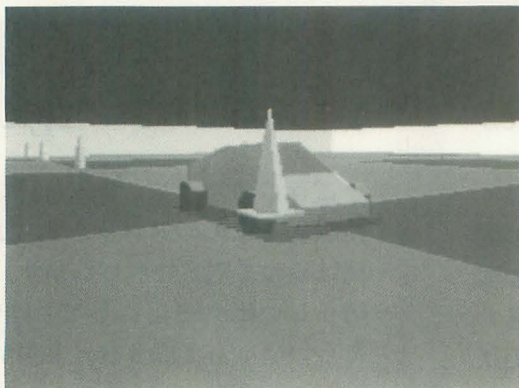
Tan Akihiko 丹 明彦

いよいよ，本格的に車の挙動をシュミレートする
まずは，いままでのシュミーレートモデルにフレームとサスペンションを組み込み
計算機の中に力学法則を持ち込むことから考えていく

「デイトナUSA」上級は相変わらず歯が立たない。というより，私の場合は精神状態が良好でなかったり体調がすぐれなかったりするとモロに走りに影響するようである。上達の兆しがないのは，このところ気分に余裕がない証拠といえるだろう。

で，ようやく出てきた「リッジレーサー2」である。私は主に中級コースを走るのだが，いや夜中のシーン(4周設定だと2周目の後半から3周目)が熱い熱い。真っ暗な峠道をロービームだけで全開走行というクレイジーさがよい。しかも車が前作よりさらに軽快に走るし，衝突してもあまり失速しなくなったみたいだし，地道ながら確実な改良の跡を感じた。実力の近い相手を見つけて対戦してみたいものだ。

＊　＊　＊

今月は自動車シミュレーションの制作を佳境に突入させるつもりだったのだが，その前段階として，車の挙動をビジュアルに表現できるシステムを作った。正確にいうなら，SLASHの扱いに意外とてこずって，グラフィックまわりを固めるだけで終わってしまったのだ。というわけで，どのような項目をシミュレートするかを固め，そのためにどの程度の表示系が必要になるかを検討してみることにしよう。

今回制作したシミュレートモデル

自然現象をシミュレートするということ

計算機でシミュレートしやすい自動車とはどんな自動車だろうか？

妙な質問に思えるが，この問いはドライビングシミュレーションゲームのある一面を見せてくれる。つまり，現実世界での自動車の機構の複雑さと，シミュレーションプログラムの複雑さはかならずしも一致しない。場合によっては逆の関係になることもあるのである。

たとえば，現代のF-1マシンは，少しでも速く走るために，想像を絶するほど複雑精巧なシステムになっている。が，ゲーム化するのはある意味で妙にやさしいのである(ほかに比べれば，だが)。それは，自動車の運動をモデル化する際にシステム設計者が呪文のように唱える「問題を簡単にするために，××の外乱要因を無視する」の類のいいわけがほとんど必要ないからである。特に1993年のF-1マシン。コンピュータ制御されたアクティブサスペンションの前では，ほとんど力学法則などあってなきがごとくである。ストレートをかっとんでもコーナーに突っ込んでも，車体は常に最適な空力特性を示し，4つの車輪は常に路面を正確に捉える。その走る姿はレールに乗っているように滑らかだが，どこか物足りなくもある。こんな自動車のシミュレーションは，車体が常に理想状態にあることを前提としていいのだから，ある程度楽ができるのである(それでも決して簡単とは思わないが)。

逆に，ごくふつうの，それも電子制御がまったく入っていないシンプルな自動車の運動をシミュレートしようとすると，計算しなくてはならない項目がそれこそ山のように出てくる。コーナリング中には遠心力のため車体は外側へ傾く(このような横揺れをローリングという)し，アクセルを開ければ慣性の作用で車体は後ろへ傾き，ブレーキを思いっきり踏

みつければ前のめりになる(前後の揺れはピッチング)。限界を超えればスピンするし，限界の直前でコントロールしタイヤを滑らせながらコーナーを立ち上がっていくドリフト走行もするだろう。自動車の機構は単純でも，これを計算機上で実現するためには，相当に複雑な力学モデルを立てることが必要になる。

なぜこうなるのか。それは，「計算機の内部の世界は力学法則の支配下にない」からである。この地上のあらゆる物体は重力の影響を免れないが，ひとたび計算機世界に入り込めば，リンゴを空中に放り出したところで落っこちはしないのである。ここで「無重力状態をシミュレートしてみました」とかいい放って上司を納得させることができるようになれば，あなたも立派な職業プログラマである(つまりバグではなくて仕様ってやつだ)が，趣味で作るプログラムくらいは，生命を吹き込んでみたいじゃないか。計算機の中に力学法則が存在しないというのであれば，我々の手で作り出せばいいのである。世界と世界を支配する法則を作る。計算機世界を支配する法則，CPUパワーとメモリ容量の制限にひっかからないように巧妙なモデルを構築できるかどうかが，プログラマの腕の見せどころなのである。

タイヤとフレームとサスペンション

自動車はさまざまな部品から構成されているが，ここでは特にタイヤとフレームとサスペンションに注目してみたい(図1)。

自動車と外界の，ほとんど唯一の接点はタイヤである。前にもいった自動車の3つの動き「走る，曲がる，止まる」のどれにもタイヤは関係する。駆動力，コーナリングフォース，そして制動力。どれもタイヤと路面の間に生じる力である。

車体は，駆動力の源であるエンジンを搭載し，制御の主体であるドライバーが搭乗する重要な部分である。が，とりあえずは，4つのタイヤがバラバラにならないように1カ所にまとめておくためのフレームとしての役割を持たせる。現実世界のフレームは，外力によるねじれやたわみなどにより走行に悪影響が出ないよう，さまざまな形状，材料で設計されるのだが，そこは計算機シミュレーションの強み，1枚板を想定しておけば十分である。実際のプログラムではその板さえも定義していない。車の各部の座標の関係さえしっかり定義しておけばよい。

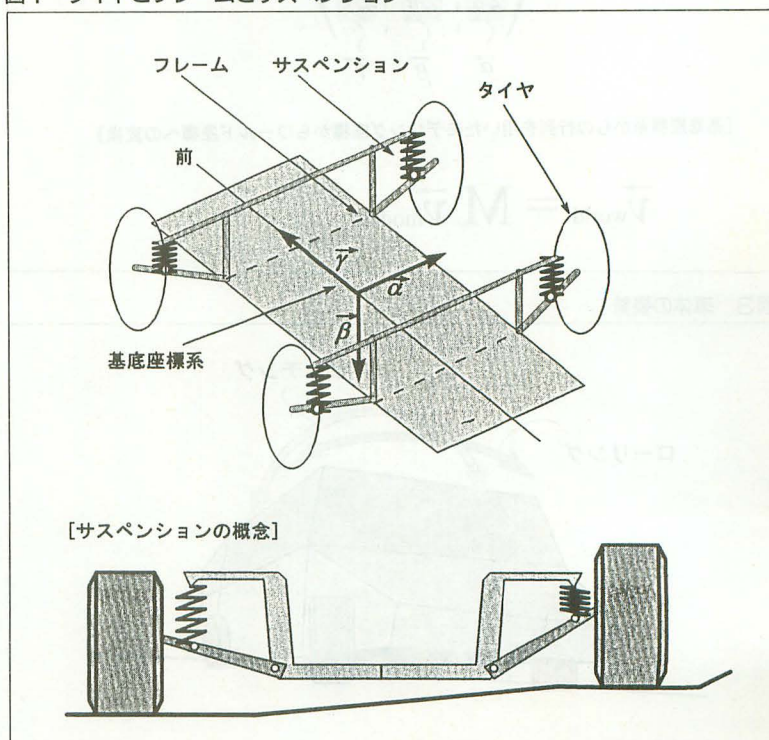
タイヤとフレームの間にあるのがサスペンションである。サスペンションは路面の凸凹から伝わるさまざまな衝撃を吸収したり，少々の段差でもタイヤが路面から浮いてしまうことのないようにタイヤを路面に押しつけたりするための装置である。概念的には，タイヤとフレームの間に設けられたバネと考えればよいだろう。単にバネだけだと，いったん起こった振動が収まらなくなるので，ダンパーという振動を鎮静化させる装置と組にされるのがふつうである。現実世界では，さまざまなサスペンション形式，つまりバネの配置の仕方やフレームをも巻き込んだ装置形状が存在するのだが，計算機シミュレーションであるから，特性の素直なバネを想定し，タイヤとフレームの位置関係をプログラムすればよいだろう。

タイヤ，フレーム，サスペンション。これらの位置関係を求めれば，自動車のグラフィック表示は可能になる。自動車の運動の力学シミュレーションも，基本的にはこの三者を中心に考えればよい。

むろん，自動車の走行特性を決定するのはこれらの座標情報だけではなく，エンジンレイアウトや駆動方式，サスペンション形式やフレーム剛性など，多くの要素がある。これらは慣性モーメントや重心，ロールセンターなどのもっと単純で扱いやすい概念に置き換える。また影響の小さい要素は無視してもよい。

具体的な表示方法

それでは車のモデルを作り，表示するまでの考え方のひとつを紹介する。しょせん計算機シミュレーションは数値の遊びにすぎないから，表示がリアル

図1　タイヤとフレームとサスペンション

であるかどうかは本質的なことではないのだが，操作するのが人間であることを考えると，視覚的なインフォメーションはとても重要になってくるのである。具体的とはいっても，できるだけSLASHの作法にとらわれない一般的な方法にしたいところである。

・基底座標系(図2)

過去に説明した内容である。3次元図形は，それが剛体である(外力によって変形したり振動したりしない)ならば，その図形をどう平行移動し，どう回転したかで完全に表現することができる。

計算上は，その図形をモデリングするのに用いた座標系(モデリング座標系)を，実際にシミュレーションを行う世界の座標系(ワールド座標系)にどう配置するかを，モデリング座標系の3軸を回転したあとの軸を3本の直行する単位ベクトルで，モデリング座標系の原点の移動量を位置ベクトルで，それぞれ表現する。言葉で説明するとかえって難しい。

基底座標の軸の取り方はSLASHにならう。向かって右がα軸(モデリング座標のX軸に対応)，下がβ軸(同じくY軸)，前方がγ軸(Z軸)である。SLASHの座標系は世間の標準からは少し外れているのだが，ここではこれに従う。独自の軸表記を用いてもいいのだろうが，座標軸の入れ替えは死ぬ思いをすることになる(経験あり)し，バグの温床にもなるので，そういう危険性はあらかじめ排除する。

また，基底座標は行列と相性がいい。というより，基底座標の3軸のベクトルを並べれば，それがそのまま回転行列になるのである。回転を行列で表現することの利点は合成することができることである。たとえばタイヤは，車体との相対的な位置関係で表現するのが楽なのだが，表示の際にはワールド座標系に対する位置関係を求めなくてはならない。このとき，「タイヤの車体に対する相対的な基底座標系」を行列で表現しておけば，これに「車体のワールド座標系に対する相対的な基底座標系」の行列を掛け合わせることで，「タイヤのワールド座標系に対する相対的な基底座標系」を求めることができる。この合成変換は，オイラー角(3軸に対する回転角で物体の回転を表現する)による表記では難しい。階層構造を持った物体の表現には基底座標系とそれに対応する行列を利用することをお勧めする。

SLASHは，バージョン2からこの行列による変換をサポートしており，基底座標系が扱いやすくなった。

・車体の姿勢(図3)

重力，遠心力，慣性力などとサスペンションの力の釣り合いから車体の姿勢は決定される。この車体の姿勢は基底座標で表現することができる。これに沿って車のボディを表示すれば，ピッチングやローリングしている様子を視覚的に表現できるというわけである。今回は力学シミュレーションをまったく行っていないので，車体の姿勢はアクセル，ブレーキおよびステアリングの操作からいいかげんに算出している。

・前進/後退に伴うタイヤの転がり

現時点で実現した自動車の運動は，過去に説明した極低速旋回と直進運動だけであるが，これだけでも一応タイヤの回転は表現可能である。視覚効果を

図2　基底座標系

[基底座標系と行列の関係]

$$M = \begin{pmatrix} x_\alpha & x_\beta & x_\gamma \\ y_\alpha & y_\beta & y_\gamma \\ z_\alpha & z_\beta & z_\gamma \end{pmatrix}$$

[基底座標系からの行列を用いたモデリング座標からワールド座標への変換]

$$\vec{v}_{world} = M \vec{v}_{model}$$

図3　車体の姿勢

狙ってのことであるから，タイヤが路面を滑ることなく転がっているように見せるため，回転量をある程度真面目に計算することにする。

極低速旋回では，タイヤごとの旋回半径(具体的には旋回中心からタイヤの接地点までの距離)と，タイヤ自身の半径の比を求め，車全体の旋回角度にかけることでタイヤ自身の回転角を求めることができる(図4)。

直進運動の場合はもっと簡単で，移動距離とタイヤの半径から回転角を求めることができる。

いずれも，回転量はタイヤごとに記憶しておく。これにより，次第に各タイヤの転がり量がばらついてくる様子が観測できる。

・ステアリングを反映したタイヤの向き

ステアリング角度は，コーナリングフォースの発生など，車の旋回運動をシミュレートする際に重要な情報となる。が，ここでは視覚的な効果のみを狙って，タイヤ形状がステアリングに合わせて回転するようにする。

今回は，計算量の増加に見合う効果が得られないという判断から，前輪の左右のタイヤでステアリング角度を微妙に変えて4輪の旋回中心を合わせるアッカーマン旋回には対応していない。

・タイヤの姿勢(図2)

以上のタイヤの転がりとステアリングによる回転から各タイヤの姿勢を求めることができる。例によって基底座標で表現する。前述したように，最終的には車体の基底座標系と合成する。

前輪はβ軸まわりの回転(ステアリング)とα軸まわりの回転(転がり)という2軸に対する回転を同時に行うため，扱いに気を遣う必要がある。注意したいのは，α軸まわりの回転を先にすること。さもないと，タイヤが斜めになってがったんごっとんと回り出すというみっともない状況に陥る。

実はデバッグ時にはもっとひどい状況もあった。車体と4輪が独立してスピンするなどという，現実では絶対にあり得ないことが起きるのもまた計算機シミュレーションならではのできごとである。

後輪は転がりだけであるからもう少し楽である。なお4WS(4輪操舵)は考慮していない。

・カメラの基底座標系(図5)

SLASHの行列による座標変換は，カメラの基底座標系の行列をパラメータとして取るものである。カメラの基底座標系とは，ワールド座標の中に配置したカメラから向かって右，下，前の方向を3軸とした座標系である。

地表以外の物体(今回は車とそのタイヤだけ)は，明示的にカメラの前に持ってこなくては，つまり合成変換によってワールド座標に持ってこなくては，正しい位置に表示されない。このため，物体の基底座標系の行列の逆行列(今回のプログラムでは，回転行列の逆行列は転置行列であるという性質を利用して計算量を減らしている)を求め，それをカメラの行列に掛けて用いている。

また物体の位置については，カメラとの相対位置にカメラの行列を掛けている。

・地表の座標変換

地表(と，地表に固定されたパイロン)の座標変換には，SLASHで「視点座標変換」と呼ばれている座標変換を用いている。これを採用したのは，シェーディングが良好になるためである。通常の変換では，視点の移動/回転につれてシェーディングが変化してしまうのだが，視点座標変換を用いて表示した物体はシェーディングが変化しないので見た目がいいのだ。

通常の座標変換が物体中心の回転を行うのに対し，視点座標変換は視点中心で回転を行う。このため，それ自身が回転運動をする車などの移動物体は制御が難しい。

図4 タイヤの転がり

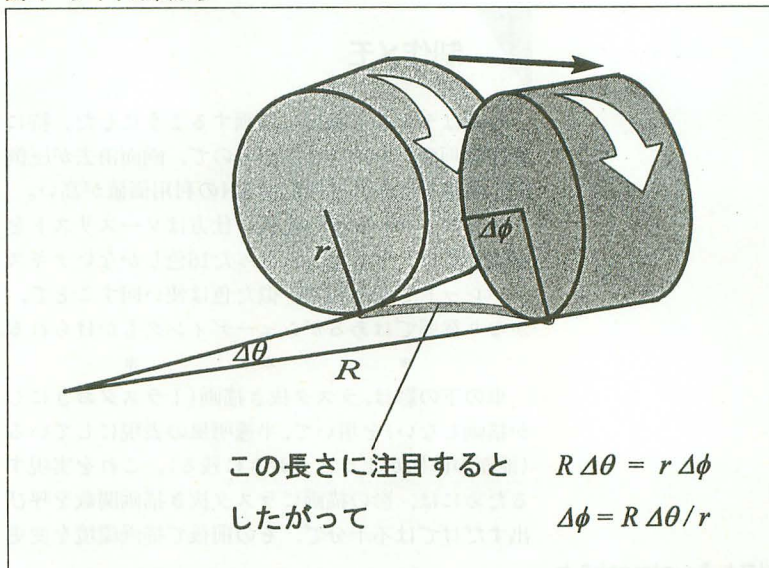
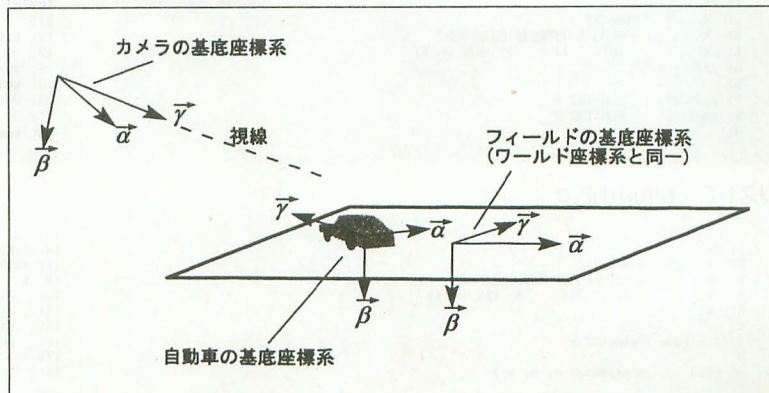



図5 カメラの基底座標系

ハードコア3Dエクスタシー(第11回)

今回のプログラム

壁に囲まれた市松模様のフィールドにパイロンを何本か立てた場所を車が走り回るだけである。まだ力学シミュレーションはまったく入っていないし，パイロンとの当たり判定もない。コーナリングをすると車体はロールするし，アクセルやブレーキではピッチングするが，これは無根拠に適当な角度で車体を傾けているだけ。パラメータを指定すればローリングやピッチングの姿勢表現ができることの確認のための，にせローリング/にせピッチングである。

操作は，

　マウスの右ボタンでアクセル
　マウスの左ボタンでブレーキ
　マウスの左右でハンドル
　F1キーで上空からの視点(もう1度押すと戻る)
　テンキーで車の周囲の視点
　スペースキーでポーズ

といったところである。

制作メモ

今回はテキスト画面に描画するようにした。特に地面は画面いっぱいに広がるので，画面消去が圧倒的に高速なテキスト版SLASHの利用価値が高い。

テキストパレットの定義の仕方はソースリストを解読していただきたい。たった16色しかないテキストパレットを割り付け，似た色は使い回すことで，かなり貧弱ではあるがシェーディングもかけられる。

*　　*　　*

車の下の影は，ラスタ抜き描画(1ラスタおきにしか描画しない)を用いて，半透明風の表現にしている(地面の模様が1ラスタおきに残る)。これを実現するためには，影の描画にラスタ抜き描画関数を呼び出すだけでは不十分で，その前後で描画環境を変更する必要がある。画面サイズや描画ページの設定などを一瞬だけラスタ抜き用に変更して，影の描画が終わった直後に設定を戻している。

*　　*　　*

車とパイロンのモデリングには坪井氏作のモデラを用いた。相当使いやすくなっているので次回の配布時には乞うご期待。その出力(PLG形式をplgconv.xでアセンブラソースに変換)に手を加えて使っている。モデラではサイズに気を遣わないで作ったので，サイズを調整するための関数pack( )を書いた。

*　　*　　*

今回も実数演算を多用しているので，X68030＋数値演算コプロセッサでないと悲惨だろう。これは計算精度で悩むのを嫌ったからで，必要な計算精度がわかれば，整数か固定小数点数に変更する。なお，上記の組み合わせでは20fps以上は常に出ている。

ちなみに，例の高速道路風ドライブシミュレータは，実数演算を多用していたブロックのソーティングをやめてテーブル化することでさらに速くなり，20fpsに届きそうな勢いである。

終わりに

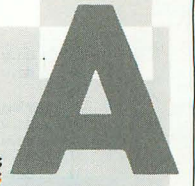
とりあえずジムカーナを目指したい気分になってきた。コースはモデリングというほどのものは必要ない。それだけに車の動きそのものが重要になっていてごまかしがきかない。FFとFRとMRの差が運転してわかるくらいに作り込まないと満足してもらえないだろう。逆にいえば，実車とあまり違和感のないジムカーナ・シミュレータが作れれば，ほかのたいていの自動車競技には対応できるに違いない。悪路でドリフトさせまくるラリーというのもいいなあ。

風呂敷を広げ始めるときりがないのでこのへんでやめておくが，とにかく私は運転そのものを楽しめるドライブシミュレーションを欲しているのだ，というわけでまた来月。

リスト1　slmath2.h

```
 1: /*
 2:  *        slmath2.h
 3:  *        - ベクトル/行列演算(倍精度実数)
 4:  *          Jul. 1994      丹 明彦(Oh!X)
 5:  */
 6:
 7: #ifndef    __SLMATH2_H__
 8: #define    __SLMATH2_H__
 9:
10: #include  <slash3/slmath.h>
11:
12: typedef double    MATRIX3[3][3];
13:
14: void        applymat2( VECTOR3, MATRIX3, VECTOR3 );
15: void        multmat2( MATRIX3, MATRIX3, MATRIX3 );
16: void        invertmat2( MATRIX3, MATRIX3 );
17:
18: #endif     /* __SLMATH2_H__ */
```

リスト2　slmath2.c

```
 1: /*
 2:  *        slmath2.c
 3:  *        - ベクトル/行列演算(倍精度実数)
 4:  *          Jul. 1994      丹 明彦(Oh!X)
 5:  */
 6:
 7: #include  "slmath2.h"
 8:
 9: void      applymat2( d, m, v )
10: VECTOR3   d, v;
11: MATRIX3   m;
12: {
13:   d[0] = m[0][0]*v[0] + m[0][1]*v[1] + m[0][2]*v[2];
14:   d[1] = m[1][0]*v[0] + m[1][1]*v[1] + m[1][2]*v[2];
15:   d[2] = m[2][0]*v[0] + m[2][1]*v[1] + m[2][2]*v[2];
16:   return;
17: }
18:
```

```
19: void       multmat2( d, m1, m2 )
20: MATRIX3    d, m1, m2;
21: {
22:   d[0][0] = m1[0][0]*m2[0][0] + m1[0][1]*m2[1][0] + m1[0][2]*m2[2][0];
23:   d[0][1] = m1[0][0]*m2[0][1] + m1[0][1]*m2[1][1] + m1[0][2]*m2[2][1];
24:   d[0][2] = m1[0][0]*m2[0][2] + m1[0][1]*m2[1][2] + m1[0][2]*m2[2][2];
25:   d[1][0] = m1[1][0]*m2[0][0] + m1[1][1]*m2[1][0] + m1[1][2]*m2[2][0];
26:   d[1][1] = m1[1][0]*m2[0][1] + m1[1][1]*m2[1][1] + m1[1][2]*m2[2][1];
27:   d[1][2] = m1[1][0]*m2[0][2] + m1[1][1]*m2[1][2] + m1[1][2]*m2[2][2];
28:   d[2][0] = m1[2][0]*m2[0][0] + m1[2][1]*m2[1][0] + m1[2][2]*m2[2][0];
29:   d[2][1] = m1[2][0]*m2[0][1] + m1[2][1]*m2[1][1] + m1[2][2]*m2[2][1];
30:   d[2][2] = m1[2][0]*m2[0][2] + m1[2][1]*m2[1][2] + m1[2][2]*m2[2][2];
31:   return;
32: }
33:
34: void       invertmat2( d, m )
35: MATRIX3    d, m;
36: {
37:   /* 回転行列と仮定しているので逆行列は転地 */
38:   d[0][0] = m[0][0]; d[0][1] = m[1][0]; d[0][2] = m[2][0];
39:   d[1][0] = m[0][1]; d[1][1] = m[1][1]; d[1][2] = m[2][1];
40:   d[2][0] = m[0][2]; d[2][1] = m[1][2]; d[2][2] = m[2][2];
41:   return;
42: }
```

リスト3 textcolor.h

```
 1: /*
 2:  *        textcolor.h
 3:  *        - テキスト色の定義
 4:  *          Jul. 1994     丹 明彦(Oh!X)
 5:  */
 6:
 7: /* 名前つきカラー(テキストSLASH用に4ビットシフト) */
 8: #define  TP_BACKGROUND    (0<<4)
 9: #define  TP_GROUND1       (1<<4)
10: #define  TP_GROUND2       (2<<4)
11: #define  TP_WALL1         (3<<4)
12: #define  TP_WALL2         (4<<4)
13: #define  TP_WALL3         (5<<4)
14: #define  TP_CAR1          (6<<4)
15: #define  TP_CAR2          (7<<4)
16: #define  TP_CAR3          (8<<4)
17: #define  TP_TIRE1         (0<<4)
18: #define  TP_TIRE2         (9<<4)
19: #define  TP_TIRE3         (10<<4)
20: #define  TP_TIRE4         (3<<4)
21: #define  TP_TIRE5         (4<<4)
22: #define  TP_TIRE6         (5<<4)
23: #define  TP_PYLON1        (11<<4)
24: #define  TP_PYLON2        (12<<4)
25: #define  TP_PYLON3        (13<<4)
26: #define  TP_PYLON4        (14<<4)
27: #define  TP_PYLON5        (15<<4)
28:
29: /* パレットコードに対応したカラー */
30: #define  TC_00   RGBI(0,0,0,0)
31: #define  TC_01   RGBI(0,0,16,0)
32: #define  TC_02   RGBI(0,16,0,0)
33: #define  TC_03   RGBI(12,12,12,0)
34: #define  TC_04   RGBI(16,16,16,0)
35: #define  TC_05   RGBI(20,20,20,0)
36: #define  TC_06   RGBI(10,0,0,0)
37: #define  TC_07   RGBI(20,0,0,0)
38: #define  TC_08   RGBI(30,0,0,0)
39: #define  TC_09   RGBI(4,4,4,0)
40: #define  TC_10   RGBI(8,8,8,0)
41: #define  TC_11   RGBI(8,8,0,0)
42: #define  TC_12   RGBI(12,12,0,0)
43: #define  TC_13   RGBI(16,16,0,0)
44: #define  TC_14   RGBI(20,20,0,0)
45: #define  TC_15   RGBI(24,24,0,0)
46:
47: /* テキスト用シェーディングテーブル */
48: #define  CARCOLORS     ¥
49:   TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,¥
50:   TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,TP_CAR1,¥
51:   TP_CAR2,TP_CAR2,TP_CAR2,TP_CAR2,TP_CAR2,TP_CAR2,TP_CAR2,TP_CAR2,¥
52:   TP_CAR3,TP_CAR3,TP_CAR3,TP_CAR3,TP_CAR3,TP_CAR3,TP_CAR3,TP_CAR3
53:
54: #define  TIRECOLORS    ¥
55:   TP_TIRE1,TP_TIRE1,TP_TIRE1,TP_TIRE1,¥
56:   TP_TIRE1,TP_TIRE1,TP_TIRE1,TP_TIRE1,¥
57:   TP_TIRE2,TP_TIRE2,TP_TIRE2,TP_TIRE2,¥
58:   TP_TIRE2,TP_TIRE2,TP_TIRE3,TP_TIRE3,¥
59:   TP_TIRE3,TP_TIRE3,TP_TIRE3,TP_TIRE3,¥
60:   TP_TIRE4,TP_TIRE4,TP_TIRE4,TP_TIRE4,¥
61:   TP_TIRE4,TP_TIRE4,TP_TIRE5,TP_TIRE5,¥
62:   TP_TIRE5,TP_TIRE5,TP_TIRE6,TP_TIRE6
63:
64: #define  PYLONCOLORS   ¥
65:   TP_PYLON1,TP_PYLON1,TP_PYLON1,TP_PYLON1,¥
66:   TP_PYLON1,TP_PYLON1,TP_PYLON1,TP_PYLON1,¥
67:   TP_PYLON1,TP_PYLON1,TP_PYLON1,TP_PYLON1,¥
68:   TP_PYLON1,TP_PYLON1,TP_PYLON1,TP_PYLON1,¥
69:   TP_PYLON2,TP_PYLON2,TP_PYLON2,TP_PYLON2,¥
70:   TP_PYLON3,TP_PYLON3,TP_PYLON3,TP_PYLON3,¥
71:   TP_PYLON4,TP_PYLON4,TP_PYLON4,TP_PYLON4,¥
72:   TP_PYLON5,TP_PYLON5,TP_PYLON5,TP_PYLON5
```

リスト4 pack.c

```
 1: /*
 2:  *       pack.c
 3:  *       - ポリゴンリストを指定の直方体領域に押し込む
 4:  *         Jul. 1994     丹 明彦(Oh!X)
 5:  */
 6:
 7: #include <slash3/slashlib.h>
 8:
 9: void       pack( SLPOINTLIST *pl,
10:          int x1, int x2,
11:          int y1, int y2,
12:          int z1, int z2 )
13: {
14:   int      xmin = 65535, xmax = -65535;
15:   int      ymin = 65535, ymax = -65535;
16:   int      zmin = 65535, zmax = -65535;
17:   int      n, i, x, y, z;
18:
19:   n = pl->n;
20:   for ( i = 0; i < n; i++ ) {
21:          x = pl->point[i].x;
22:          if ( x < xmin ) xmin = x;
23:          if ( x > xmax ) xmax = x;
24:          y = pl->point[i].y;
25:          if ( y < ymin ) ymin = y;
26:          if ( y > ymax ) ymax = y;
27:          z = pl->point[i].z;
28:          if ( z < zmin ) zmin = z;
29:          if ( z > zmax ) zmax = z;
30:   }
31:   for ( i = 0; i < n; i++ ) {
32:          pl->point[i].x = x1 + (pl->point[i].x - xmin)*(x2-x1)/(xmax-xmin);
33:          pl->point[i].y = y1 + (pl->point[i].y - ymin)*(y2-y1)/(ymax-ymin);
34:          pl->point[i].z = z1 + (pl->point[i].z - zmin)*(z2-z1)/(zmax-zmin);
35:   }
36:   return;
37: }
```

リスト5 pylon.s

```
 1: Tri        equ    0
 2: Quad       equ    1
 3: Line       equ    2
 4:
 5:    .xdef   _pylon_pointlist
 6:    .xdef   _pylon_polygonlist
 7:
 8: _pylon_pointlist:
 9:    dc.w    17
10:    dc.w    392,0,-392
11:    dc.w    392,0,392
12:    dc.w    392,16,392
13:    dc.w    392,16,-392
14:    dc.w    -392,0,392
15:    dc.w    -392,16,392
16:    dc.w    -392,0,-392
17:    dc.w    -392,16,-392
18:    dc.w    209,0,-211
19:    dc.w    0,-755,0
20:    dc.w    275,0,0
21:    dc.w    0,0,-281
22:    dc.w    -213,0,-213
23:    dc.w    -281,0,0
24:    dc.w    -211,0,209
```

```
25:  dc.w    0,0,275
26:  dc.w    207,0,207
27:
28: _pylon_polygonlist:
29:  dc.w    14
30:  dc.w    Quad
31:  dc.w    4,1,0,6
32:  dc.w    0,0
33:  dc.l    std_yellow
34:  ds.w    7
35:  dc.w    Tri
36:  dc.w    11,9,8,0
37:  dc.w    0,0
38:  dc.l    std_yellow
39:  ds.w    7
40:  dc.w    Tri
41:  dc.w    8,9,10,0
42:  dc.w    0,0
43:  dc.l    std_yellow
44:  ds.w    7
45:  dc.w    Tri
46:  dc.w    16,9,15,0
47:  dc.w    0,0
48:  dc.l    std_yellow
49:  ds.w    7
```

```
50:  dc.w    Tri
51:  dc.w    10,9,16,0
52:  dc.w    0,0
53:  dc.l    std_yellow
54:  ds.w    7
55:  dc.w    Tri
56:  dc.w    14,9,13,0
57:  dc.w    0,0
58:  dc.l    std_yellow
59:  ds.w    7
60:  dc.w    Tri
61:  dc.w    15,9,14,0
62:  dc.w    0,0
63:  dc.l    std_yellow
64:  ds.w    7
65:  dc.w    Tri
66:  dc.w    13,9,12,0
67:  dc.w    0,0
68:  dc.l    std_yellow
69:  ds.w    7
70:  dc.w    Tri
71:  dc.w    12,9,11,0
72:  dc.w    0,0
73:  dc.l    std_yellow
74:  ds.w    7
```

```
75:  dc.w    Quad
76:  dc.w    0,1,2,3
77:  dc.w    0,0
78:  dc.l    std_yellow
79:  ds.w    7
80:  dc.w    Quad
81:  dc.w    1,4,5,2
82:  dc.w    0,0
83:  dc.l    std_yellow
84:  ds.w    7
85:  dc.w    Quad
86:  dc.w    4,6,7,5
87:  dc.w    0,0
88:  dc.l    std_yellow
89:  ds.w    7
90:  dc.w    Quad
91:  dc.w    6,0,3,7
92:  dc.w    0,0
93:  dc.l    std_yellow
94:  ds.w    7
95:  dc.w    Quad
96:  dc.w    2,5,7,3
97:  dc.w    0,0
98:  dc.l    std_yellow
99:  ds.w    7
```

リスト6　tcourse.c

```
  1: /*
  2:   tcourse.c
  3:   - テストコース(テキスト画面)
  4:     Jul. 1994     丹 明彦(Oh!X)
  5: */
  6:
  7: #include  <stdlib.h>
  8: #include  <slash3/slashlib.h>
  9: #include  <slash3/addprim.h>
 10: #include  <slash3/stdcolor.h>
 11: #include  <slash3/slmath.h>
 12: #include  "textcolor.h"
 13:
 14: SLPOLYGONLIST      *field_polygonlist;
 15: SLPOINTLIST        *field_pointlist;
 16:
 17: SLPOLYGONLIST      *collide_polygonlist;
 18: SLPOINTLIST        *collide_pointlist;
 19:
 20: #define   COLLIDE_TOLERANCE       25
 21:
 22: extern SLPOLYGONLIST      pylon_polygonlist;
 23: extern SLPOINTLIST        pylon_pointlist;
 24: extern int        npylon;
 25: extern IVECTOR    pylon[100];
 26:
 27: /* フィールド */
 28: void      create_field()
 29: {
 30:   int     x, z, i, j, n;
 31:
 32: #define   FS      8192            /* フィールドのサイズ */
 33: #define   FD      8               /* フィールドの分割数 */
 34: #define   WH      512             /* 壁の高さ */
 35:
 36:   field_polygonlist = malloc( sizeof(SLPOLYGONLIST)+sizeof(SLPOLYGON)*400 );
 37:   field_pointlist = malloc( sizeof(SLPOINTLIST)+sizeof(SLPOINT)*1600 );
 38:   field_polygonlist->n = 0;
 39:   field_pointlist->n = 0;
 40:
 41:   collide_polygonlist = malloc( sizeof(SLPOLYGONLIST)+sizeof(SLPOLYGON)*400 );
 42:   collide_pointlist = malloc( sizeof(SLPOINTLIST)+sizeof(SLPOINT)*1600 );
 43:   collide_polygonlist->n = 0;
 44:   collide_pointlist->n = 0;
 45:
 46:   /* パイロンの位置決め */
 47:   npylon = 0;
 48:   for ( i = 1; i <= FD; i++ ) {
 49:           pylon[i][0] = 0;
 50:           pylon[i][1] = 0;
 51:           pylon[i][2] = -FS + i*(FS*2/FD);
 52:           npylon++;
 53:   }
 54:
 55:   /* 床 */
 56:   for ( i = 0; i < FD; i++ ) {
 57:           x = -FS + i*(FS*2/FD);
 58:           for ( j = 0; j < FD; j++ ) {
 59:                   z = -FS + j*(FS*2/FD);
 60:                   n = addtetragon( field_polygonlist, field_pointlist,
 61:                           x,            0, z,
 62:                           x,            0, z+(FS*2/FD),
 63:                           x+(FS*2/FD), 0, z+(FS*2/FD),
 64:                           x+(FS*2/FD), 0, z,
 65:                           ((i+j)%2)?(&std_darkblue):(&std_darkgreen) );
 66:                   noshade( field_polygonlist, n, ((i+j)%2)?(TP_GROUND1):(TP_GROUND2) );
 67:                   n = addtetragon( collide_polygonlist, collide_pointlist,
 68:                           x,            0, z,
 69:                           x,            0, z+(FS*2/FD),
 70:                           x+(FS*2/FD), 0, z+(FS*2/FD),
 71:                           x+(FS*2/FD), 0, z,
 72:                           ((i+j)%2)?(&std_darkblue):(&std_darkgreen) );
 73:                   noshade( collide_polygonlist, n, ((i+j)%2)?(TP_GROUND1):(TP_GROUND2) );
 74:           }
 75:   }
 76:
 77:   /* 壁 */
 78:   x = -FS+(FS/FD);          /* ブロックサイズの半分 */
 79:   /*for ( j = 0; j < FD; j++ ) {*/
 80:   for ( j = 0; j < FD-1; j++ ) {
 81:           /*z = -FS + j*(FS*2/FD);*/
 82:           z = -FS + j*(FS*2/FD) + (FS/FD);          /* 半ずらし */
 83:           n = addtetragon( field_polygonlist, field_pointlist,
 84:                   x,   0, z,
 85:                   x, -WH, z,
 86:                   x, -WH, z+(FS*2/FD),
 87:                   x,   0, z+(FS*2/FD),
 88:                   &std_lightgray );
 89:           noshade( field_polygonlist, n, TP_WALL2 );
 90:   }
 91:   x = FS-(FS/FD); /* ブロックサイズの半分 */
 92:   /*for ( j = 0; j < FD; j++ ) {*/
 93:   for ( j = 0; j < FD-1; j++ ) {
 94:           /*z = -FS + j*(FS*2/FD);*/
 95:           z = -FS + j*(FS*2/FD) + (FS/FD);          /* 半ずらし */
 96:           n = addtetragon( field_polygonlist, field_pointlist,
 97:                   x, -WH, z,
 98:                   x,   0, z,
 99:                   x,   0, z+(FS*2/FD),
100:                   x, -WH, z+(FS*2/FD),
101:                   &std_lightgray );
102:           noshade( field_polygonlist, n, TP_WALL2 );
103:   }
104:   z = -FS+(FS/FD);          /* ブロックサイズの半分 */
105:   /*for ( i = 0; i < FD; i++ ) {*/
106:   for ( i = 0; i < FD-1; i++ ) {
107:           /*x = -FS + i*(FS*2/FD);*/
108:           x = -FS + i*(FS*2/FD) + (FS/FD);          /* 半ずらし */;
109:           n = addtetragon( field_polygonlist, field_pointlist,
110:                   x,            -WH, z,
111:                   x,              0, z,
112:                   x+(FS*2/FD),    0, z,
113:                   x+(FS*2/FD), -WH, z,
114:                   &std_lightgray );
115:           noshade( field_polygonlist, n, TP_WALL1 );
116:   }
117:   z = FS-(FS/FD); /* ブロックサイズの半分 */
118:   /*for ( i = 0; i < FD; i++ ) {*/
119:   for ( i = 0; i < FD-1; i++ ) {
120:           /*x = -FS + i*(FS*2/FD);*/
121:           x = -FS + i*(FS*2/FD) + (FS/FD);          /* 半ずらし */
122:           n = addtetragon( field_polygonlist, field_pointlist,
123:                   x,              0, z,
124:                   x,            -WH, z,
125:                   x+(FS*2/FD), -WH, z,
126:                   x+(FS*2/FD),    0, z,
127:                   &std_lightgray );
128:           noshade( field_polygonlist, n, TP_WALL3 );
129:   }
130:
131:   /* 場外 */
132:   x = -FS;
133:   for ( j = 0; j < FD; j++ ) {
134:           z = -FS + j*(FS*2/FD);
135:           n = addtetragon( collide_polygonlist, collide_pointlist,
136:                   x-COLLIDE_TOLERANCE,           0, z-COLLIDE_TOLERANCE,
137:                   x-COLLIDE_TOLERANCE,           0, z+(FS*2/FD)+COLLIDE_TOLERANCE,
138:                   x+(FS/FD)+COLLIDE_TOLERANCE, 0, z+(FS*2/FD)+COLLIDE_TOLERANCE,  /* ブロッ
クサイズの半分 */
139:                   x+(FS/FD)+COLLIDE_TOLERANCE, 0, z-COLLIDE_TOLERANCE,            /* ブロッ
クサイズの半分 */
140:                   &std_magenta );
141:           noshade( collide_polygonlist, n, TP_BACKGROUND );
142:   }
143:   x = FS-(FS/FD); /* ブロックサイズの半分 */
144:   for ( j = 0; j < FD; j++ ) {
145:           z = -FS + j*(FS*2/FD);
146:           n = addtetragon( collide_polygonlist, collide_pointlist,
147:                   x-COLLIDE_TOLERANCE,           0, z-COLLIDE_TOLERANCE,
148:                   x-COLLIDE_TOLERANCE,           0, z+(FS*2/FD)+COLLIDE_TOLERANCE,
149:                   x+(FS/FD)+COLLIDE_TOLERANCE, 0, z+(FS*2/FD)+COLLIDE_TOLERANCE,  /* ブロッ
クサイズの半分 */
150:                   x+(FS/FD)+COLLIDE_TOLERANCE, 0, z-COLLIDE_TOLERANCE,            /* ブロッ
クサイズの半分 */
151:                   &std_magenta );
152:           noshade( collide_polygonlist, n, TP_BACKGROUND );
153:   }
154:   z = -FS;
155:   for ( i = 0; i < FD; i++ ) {
156:           x = -FS + i*(FS*2/FD);
157:           n = addtetragon( collide_polygonlist, collide_pointlist,
158:                   x-COLLIDE_TOLERANCE,             0, z-COLLIDE_TOLERANCE,
159:                   x-COLLIDE_TOLERANCE,             0, z+(FS/FD)+COLLIDE_TOLERANCE,  /* ブロッ
クサイズの半分 */
160:                   x+(FS*2/FD)+COLLIDE_TOLERANCE, 0, z+(FS/FD)+COLLIDE_TOLERANCE,  /* ブロッ
クサイズの半分 */
161:                   x+(FS*2/FD)+COLLIDE_TOLERANCE, 0, z-COLLIDE_TOLERANCE,
162:                   &std_magenta );
163:           noshade( collide_polygonlist, n, TP_BACKGROUND );
164:   }
165:   z = FS-(FS/FD); /* ブロックサイズの半分 */
166:   for ( i = 0; i < FD; i++ ) {
```


▶マンデル君にひかれるだけでは飽き足らず，X68030まで買ってしまい，危険な人どころか奇特な人となってしまいました。　中村　隆喜(21)大阪府

```
167:            x = -FS + i*(FS*2/FD);
168:            n = addtetragon( collide_polygonlist, collide_pointlist,
169:                    x-COLLIDE_TOLERANCE,            0, z-COLLIDE_TOLERANCE,
170:                    x-COLLIDE_TOLERANCE,            0, z+(FS/FD)+COLLIDE_TOLERANCE,  /* ブロッ
クサイズの半分 */
171:                    x+(FS*2/FD)+COLLIDE_TOLERANCE, 0, z+(FS/FD)+COLLIDE_TOLERANCE,  /* ブロッ
クサイズの半分 */
172:                    x+(FS*2/FD)+COLLIDE_TOLERANCE, 0, z-COLLIDE_TOLERANCE,
173:                    &std_magenta );
174:            noshade( collide_polygonlist, n, TP_BACKGROUND );
175:    }
176:    return;
177: }
178:
179: void    destroy_field()
180: {
181:    free( field_polygonlist );
182:    free( field_pointlist );
183:    free( collide_polygonlist );
184:    free( collide_pointlist );
185:    return;
186: }
```

リスト7 car.s

```
  1: Tri      equ     0
  2: Quad     equ     1
  3: Line     equ     2
  4:
  5:  .xdef   _car_pointlist
  6:  .xdef   _car_polygonlist
  7:
  8: _car_pointlist:
  9:   dc.w   18
 10:   dc.w   127,-319,63
 11:   dc.w   -128,-319,63
 12:   dc.w   -207,-143,239
 13:   dc.w   207,-143,239
 14:   dc.w   -175,64,512
 15:   dc.w   -176,-15,512
 16:   dc.w   -240,63,175
 17:   dc.w   175,64,512
 18:   dc.w   239,63,175
 19:   dc.w   -240,64,-551
 20:   dc.w   -239,-75,-543
 21:   dc.w   239,-75,-543
 22:   dc.w   240,64,-551
 23:   dc.w   -207,-175,-575
 24:   dc.w   207,-175,-575
 25:   dc.w   176,-15,512
 26:   dc.w   127,-303,-303
 27:   dc.w   -127,-303,-303
 28:
 29: _car_polygonlist:
 30:   dc.w   19
 31:   dc.w   Quad
 32:   dc.w   0,1,2,3
 33:   dc.w   0,0
 34:   dc.l   std_red
 35:   ds.w   7
 36:   dc.w   Quad
 37:   dc.w   4,5,2,6
 38:   dc.w   0,0
 39:   dc.l   std_red
 40:   ds.w   7
 41:   dc.w   Quad
 42:   dc.w   7,4,6,8
 43:   dc.w   0,0
 44:   dc.l   std_red
 45:   ds.w   7
 46:   dc.w   Quad
 47:   dc.w   9,10,11,12
 48:   dc.w   0,0
 49:   dc.l   std_red
 50:   ds.w   7
 51:   dc.w   Tri
 52:   dc.w   13,10,6,0
 53:   dc.w   0,0
 54:   dc.l   std_red
 55:   ds.w   7
 56:   dc.w   Tri
 57:   dc.w   8,11,14,0
 58:   dc.w   0,0
 59:   dc.l   std_red
 60:   ds.w   7
 61:   dc.w   Quad
 62:   dc.w   5,15,3,2
 63:   dc.w   0,0
 64:   dc.l   std_red
 65:   ds.w   7
 66:   dc.w   Quad
 67:   dc.w   16,0,3,14
 68:   dc.w   0,0
 69:   dc.l   std_red
 70:   ds.w   7
 71:   dc.w   Tri
 72:   dc.w   14,3,8,0
 73:   dc.w   0,0
 74:   dc.l   std_red
 75:   ds.w   7
 76:   dc.w   Tri
 77:   dc.w   13,6,2,0
 78:   dc.w   0,0
 79:   dc.l   std_red
 80:   ds.w   7
 81:   dc.w   Quad
 82:   dc.w   14,11,10,13
 83:   dc.w   0,0
 84:   dc.l   std_red
 85:   ds.w   7
 86:   dc.w   Quad
 87:   dc.w   12,8,6,9
 88:   dc.w   0,0
 89:   dc.l   std_red
 90:   ds.w   7
 91:   dc.w   Quad
 92:   dc.w   15,7,8,3
 93:   dc.w   0,0
 94:   dc.l   std_red
 95:   ds.w   7
 96:   dc.w   Quad
 97:   dc.w   4,7,15,5
 98:   dc.w   0,0
 99:   dc.l   std_red
100:   ds.w   7
101:   dc.w   Quad
102:   dc.w   17,1,0,16
103:   dc.w   0,0
104:   dc.l   std_red
105:   ds.w   7
106:   dc.w   Quad
107:   dc.w   1,17,13,2
108:   dc.w   0,0
109:   dc.l   std_red
110:   ds.w   7
111:   dc.w   Quad
112:   dc.w   17,16,14,13
113:   dc.w   0,0
114:   dc.l   std_red
115:   ds.w   7
116:   dc.w   Tri
117:   dc.w   8,12,11,0
118:   dc.w   0,0
119:   dc.l   std_red
120:   ds.w   7
121:   dc.w   Tri
122:   dc.w   9,6,10,0
123:   dc.w   0,0
124:   dc.l   std_red
125:   ds.w   7
```

リスト8 tdrive.c

```
 1: /*
 2:  *         tdrive.c
 3:  *         - 車の動作(テキスト画面)
 4:  *           Jul. 1994      丹 明彦(Oh!X)
 5:  */
 6:
 7: #define             __IOCS_INLINE__
 8: #include  <iocslib.h>
 9: #define             __DOS_INLINE__
10: #include  <doslib.h>
11: #include  <stdio.h>
12: #include  <stdlib.h>
13:
14: #include  <slash3/slashlib.h>
15: #include  <slash3/addprim.h>
16: #include  <slash3/check.h>
17: #include  <slash3/timedifference.h>
18: #include  "slmath2.h"
19: #include  <slash3/perfmon.h>
20: #include  <slash3/doublebuffer.h>
21: #include  "textcolor.h"
22:
23: #define   N_POINT          800
24: #define   N_OBJECT         4
25:
26: unsigned long     CcarT[32]   = { CARCOLORS   };
27: unsigned long     CtireT[32]  = { TIRECOLORS  };
28: unsigned long     CpylonT[32] = { PYLONCOLORS };
29:
30: extern SLPOLYGONLIST       *field_polygonlist;
31: extern SLPOINTLIST         *field_pointlist;
32:
33: extern SLPOLYGONLIST       *collide_polygonlist;
34: extern SLPOINTLIST         *collide_pointlist;
35:
36: void      create_field();
37: void      destroy_field();
38:
39: extern SLPOLYGONLIST       car_polygonlist;
40: extern SLPOINTLIST         car_pointlist;
41: SLPOLYGONLIST              *fwheel_polygonlist;
42: SLPOINTLIST                *fwheel_pointlist;
43: extern SLPOLYGONLIST       pylon_polygonlist;
44: extern SLPOINTLIST         pylon_pointlist;
45: SLPOLYGONLIST              *shadow_polygonlist;
46: SLPOINTLIST                *shadow_pointlist;
47:
48: SLTRANSWORK       *work;
49: SLPARAMETRR       parameter;
50:
51: typedef struct {
52:   double  width;          /* ボディ全幅 */
53:   double  length;         /* ボディ全長 */
54:   double  height;         /* ボディ全高 */
55:   double  fshaft;         /* Fホイール幅 */
56:   double  rshaft;         /* Rホイール幅 */
57:   double  fwidth;         /* Fタイヤ幅 */
58:   double  fradius;        /* Fタイヤ半径 */
59:   double  rwidth;         /* Rタイヤ幅 */
60:   double  rradius;        /* Rタイヤ半径 */
61:   double  wheelbase;      /* ホイールベース */
62:   double  steeringratio;  /* ステアリング係数(マウスX座標と回転角の関係) */
63:   double  accel;          /* 加速度 */
64:   double  brake;          /* 減速度 */
65: } CarInfo;
66:
67: #define   RUNFIX  0.2     /* 速度の補正値(大きくするほど加速減速が極端になる) */
68:
69: extern void       pack( SLPOINTLIST*, int, int, int, int, int, int );
70:
71: VECTOR3   v, va, vb, vc;                   /* 視点の位置と基底ベクトル */
72: VECTOR3   b[2], ba[2], bb[2], bc[2];       /* 車体の位置と基底ベクトル */
73: VECTOR3   w, wa, wb, wc;                   /* 前輪の位置と基底ベクトル */
74: VECTOR3   l;                               /* 光源の方向ベクトル */
75:
76: double    radius, centerx, centery, centerz, run, co, si;
77: int       sp, time = 1;
78: int       mscur, x, y, msdt, lb, rb;
79: double    theta = 0.0;            /* ステアリング角 */
80: double    phi = 0.0;              /* 転がり角 */
81: double    phi_fr = 0.0;           /* 転がり角 */
82: double    phi_fl = 0.0;           /* 転がり角 */
83: double    phi_rr = 0.0;           /* 転がり角 */
84: double    phi_rl = 0.0;           /* 転がり角 */
85: double    vtheta, vphi, vheight, vlength;
86: int       t1, t2, dt;             /* 時刻と時差 */
87: int       velocity = 0;           /* 速度 */
88: int       viewmode = 1;           /* 視点モード(デフォルトは車内) */
89: int       viewdirection = 5;
90: int       lalpha, lbeta, ldata;
91: int       p = 0;                  /* phase */
92:
```

▶今日もまたOh!Xの読者をひとり増やしてしまった。ふっ，ふっ，ふっ。
大河原　正樹(20)福島県

```
 93: CHECKINFOLIST      *collide_checkinfolist;
 94:
 95: /* 作業用行列とベクトル */
 96: static SLMATRIX   mat;
 97: static MATRIX3    m0, m0b, m1, m1b, m2, m3, m4;
 98: static VECTOR3    v1, v2, v3;
 99:
100: /* にせローリング/にせピッチング */
101: double    roll = 0.0;
102: double    pitch = 0.0;
103: #define   ROLL    (3.0*M_PI/180)
104: #define   PITCH   (1.5*M_PI/180)
105:
106: /* パイロン */
107: int       npylon;          /* 数 */
108: IVECTOR   pylon[100];      /* 位置 */
109: char      pylondc[100];    /* 表示制御用フラグ */
110:
111: /* 車の情報 */
112: static CarInfo    testcar = {
113:   200,    /* ボディ全幅 */
114:   400,    /* ボディ全長 */
115:   120,    /* ボディ全高 */
116:   230,    /* Fホイール幅 */
117:   230,    /* Rホイール幅 */
118:   30,     /* Fタイヤ幅 */
119:   30,     /* Fタイヤ半径 */
120:   30,     /* Rタイヤ幅 */
121:   30,     /* Rタイヤ半径 */
122:   250,    /* ホイールベース */
123:   2,      /* ステアリング係数 */
124:   1,      /* 加速度 */
125:   2       /* 減速度 */
126: };
127:
128: void      create_object( c )
129: CarInfo   *c;
130: {
131:   int     i;
132:
133:   /* パイロン(モデラで作ったのでサイズ調整) */
134:   pack( &pylon_pointlist, -50, 50, -200, 0, -50, 50 );
135:   for ( i = 0; i < pylon_polygonlist.n; i++ ) {
136:           setshade( &pylon_polygonlist, i, 0, 0, (SLPALET*)&CpylonT );
137:   }
138:   AddNorm( &pylon_polygonlist, &pylon_pointlist );
139:
140:   /* 車の影 */
141:   shadow_polygonlist = malloc( sizeof(SLPOLYGONLIST)+sizeof(SLPOLYGON)*5 );
142:   shadow_pointlist = malloc( sizeof(SLPOINTLIST)+sizeof(SLPOINT)*20 );
143:   shadow_polygonlist->n = 0;
144:   shadow_pointlist->n = 0;
145:   i = addtetragon( shadow_polygonlist, shadow_pointlist,
146:            (int)(c->width/2 + c->fwidth), 0,  (int)(c->length/2),
147:            (int)(c->width/2 + c->fwidth), 0, -(int)(c->length/2),
148:           -(int)(c->width/2 + c->fwidth), 0, -(int)(c->length/2),
149:           -(int)(c->width/2 + c->fwidth), 0,  (int)(c->length/2),
150:           (SLPALET*)&CtireT );
151:   noshade( shadow_polygonlist, i, TP_BACKGROUND );
152:
153:   /* 車体(モデラで作ったのでサイズ調整) */
154:   pack( &car_pointlist, (int)(-c->width/2), (int)(c->width/2),
155:           (int)(-c->height-c->fradius*0.5), (int)(-c->fradius*0.5),
156:           (int)(-c->length/2), (int)(c->length/2) );
157:   for ( i = 0; i < car_polygonlist.n; i++ ) {
158:           setshade( &car_polygonlist, i, 0, 0, (SLPALET*)&CcarT );
159:   }
160:   AddNorm( &car_polygonlist, &car_pointlist );
161:
162:   /* 前輪(車体から独立しているため別のポリゴンリストに入れる) */
163:   fwheel_polygonlist = malloc( sizeof(SLPOLYGONLIST)+sizeof(SLPOLYGON)*14 );
164:   fwheel_pointlist = malloc( sizeof(SLPOINTLIST)+sizeof(SLPOINT)*16 );
165:   fwheel_polygonlist->n = 0;
166:   fwheel_pointlist->n = 0;
167:
168: /*   タイヤのy,z座標
169:      x軸負方向から見て時計回り
170:      1  2
171:    7/□\3
172:     □□□
173:    8\□/4
174:      6  5          */
175:
176: #define   T1      -(int)(c->fradius*1.0),  (int)(c->fradius*0.4)
177: #define   T2      -(int)(c->fradius*1.0), -(int)(c->fradius*0.4)
178: #define   T3      -(int)(c->fradius*0.4), -(int)(c->fradius*1.0)
179: #define   T4       (int)(c->fradius*0.4), -(int)(c->fradius*1.0)
180: #define   T5       (int)(c->fradius*1.0), -(int)(c->fradius*0.4)
181: #define   T6       (int)(c->fradius*1.0),  (int)(c->fradius*0.4)
182: #define   T7       (int)(c->fradius*0.4),  (int)(c->fradius*1.0)
183: #define   T8      -(int)(c->fradius*0.4),  (int)(c->fradius*1.0)
184:
185:   /* タイヤ裏 */
186:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
187:           -(int)(c->fwidth/2), T1,
188:           -(int)(c->fwidth/2), T2,
189:           -(int)(c->fwidth/2), T3,
190:           -(int)(c->fwidth/2), T8,
191:           (SLPALET*)&CtireT );
192:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
193:           -(int)(c->fwidth/2), T3,
194:           -(int)(c->fwidth/2), T4,
195:           -(int)(c->fwidth/2), T7,
196:           -(int)(c->fwidth/2), T8,
197:           (SLPALET*)&CtireT );
198:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
199:           -(int)(c->fwidth/2), T4,
200:           -(int)(c->fwidth/2), T5,
201:           -(int)(c->fwidth/2), T6,
202:           -(int)(c->fwidth/2), T7,
203:           (SLPALET*)&CtireT );
204:
205:   /* タイヤトレッド */
206:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
207:           -(int)(c->fwidth/2), T2,
208:           -(int)(c->fwidth/2), T1,
209:            (int)(c->fwidth/2), T1,
210:            (int)(c->fwidth/2), T2,
211:           (SLPALET*)&CtireT );
212:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
213:           -(int)(c->fwidth/2), T3,
214:           -(int)(c->fwidth/2), T2,
215:            (int)(c->fwidth/2), T2,
216:            (int)(c->fwidth/2), T3,
217:           (SLPALET*)&CtireT );
218:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
219:           -(int)(c->fwidth/2), T4,
220:           -(int)(c->fwidth/2), T3,
221:            (int)(c->fwidth/2), T3,
222:            (int)(c->fwidth/2), T4,
223:           (SLPALET*)&CtireT );
224:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
225:           -(int)(c->fwidth/2), T5,
226:           -(int)(c->fwidth/2), T4,
227:            (int)(c->fwidth/2), T4,
228:            (int)(c->fwidth/2), T5,
229:           (SLPALET*)&CtireT );
230:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
231:           -(int)(c->fwidth/2), T6,
232:           -(int)(c->fwidth/2), T5,
233:            (int)(c->fwidth/2), T5,
234:            (int)(c->fwidth/2), T6,
235:           (SLPALET*)&CtireT );
236:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
237:           -(int)(c->fwidth/2), T7,
238:           -(int)(c->fwidth/2), T6,
239:            (int)(c->fwidth/2), T6,
240:            (int)(c->fwidth/2), T7,
241:           (SLPALET*)&CtireT );
242:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
243:           -(int)(c->fwidth/2), T8,
244:           -(int)(c->fwidth/2), T7,
245:            (int)(c->fwidth/2), T7,
246:            (int)(c->fwidth/2), T8,
247:           (SLPALET*)&CtireT );
248:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
249:           -(int)(c->fwidth/2), T1,
250:           -(int)(c->fwidth/2), T8,
251:            (int)(c->fwidth/2), T8,
252:            (int)(c->fwidth/2), T1,
253:           (SLPALET*)&CtireT );
254:
255:   /* タイヤ表 */
256:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
257:            (int)(c->fwidth/2), T2,
258:            (int)(c->fwidth/2), T1,
259:            (int)(c->fwidth/2), T8,
260:            (int)(c->fwidth/2), T3,
261:           (SLPALET*)&CtireT );
262:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
263:            (int)(c->fwidth/2), T4,
264:            (int)(c->fwidth/2), T3,
265:            (int)(c->fwidth/2), T8,
266:            (int)(c->fwidth/2), T7,
267:           (SLPALET*)&CtireT );
268:   addtetragon( fwheel_polygonlist, fwheel_pointlist,
269:            (int)(c->fwidth/2), T5,
270:            (int)(c->fwidth/2), T4,
271:            (int)(c->fwidth/2), T7,
272:            (int)(c->fwidth/2), T6,
273:           (SLPALET*)&CtireT );
274:
275:   AddNorm( fwheel_polygonlist, fwheel_pointlist );
276:   return;
277: }
278:
279: void      destroy_object()
280: {
281:   free( fwheel_polygonlist );
282:   free( fwheel_pointlist );
283:   free( shadow_polygonlist );
284:   free( shadow_pointlist );
285:   return;
286: }
287:
288: void      drive()
289: {
290:   double  wd1, wd2;
291:
292:   /* 前回の時刻との差 */
293:   t2 = ONTIME();
294:   dt = TIMEDIFFERENCE(t2,t1);
295:   t1 = t2;
296:   /* 速度 */
297:   if ( lb && rb ) {
298:           pitch = PITCH;                  /* にせ */
299:           velocity -= testcar.brake;
300:   } else if ( rb ) {
301:           pitch = -PITCH;                 /* にせ */
302:           velocity += testcar.accel;
303:   } else if ( lb ) {
304:           if ( velocity > 0 ) {
305:                   pitch = PITCH;                  /* にせ */
306:                   velocity -= testcar.brake;
307:                   if ( velocity < 0 ) velocity = 0;
308:           } else if ( velocity < 0 ) {
309:                   pitch = 0.0;                    /* にせ */
310:                   velocity += testcar.brake;
311:                   if ( velocity > 0 ) velocity = 0;
312:           } else {
313:                   pitch = 0.0;                    /* にせ */
314:           }
315:   } else {
316:           pitch = 0.0;                    /* にせ */
317:   }
318:
319:   /* 速度に前回の時刻との差をかければ移動量が出る */
320:   run = (double)(velocity*dt)*RUNFIX;
321:
322:   theta = ITOD((128-x)*testcar.steeringratio);    /* ステアリング角 */
323:   if ( velocity == 0 ) {
324:           roll = 0.0;                     /* にせ */
325:            b[1-p][0] =  b[p][0];  b[1-p][1] =  b[p][1];  b[1-p][2] =  b[p][2];
326:           ba[1-p][0] = ba[p][0]; ba[1-p][1] = ba[p][1]; ba[1-p][2] = ba[p][2];
327:           bb[1-p][0] = bb[p][0]; bb[1-p][1] = bb[p][1]; bb[1-p][2] = bb[p][2];
328:           bc[1-p][0] = bc[p][0]; bc[1-p][1] = bc[p][1]; bc[1-p][2] = bc[p][2];
329:   } else {
330:           /* 基底座標系のγα平面上の極低速円旋回 */
331:           if ( x < 128 ) {        /* 左折 */
332:                   roll = -ROLL;                   /* にせ */
333:                   radius = testcar.wheelbase / tan( theta );     /* 旋回半径 */
334:                   centerx = b[p][0] - bc[p][0]*testcar.wheelbase/2        /* 旋回中心 */
335:                                     - ba[p][0]*radius;
336:                   centery = b[p][1] - bc[p][1]*testcar.wheelbase/2
337:                                     - ba[p][1]*radius;
338:                   centerz = b[p][2] - bc[p][2]*testcar.wheelbase/2
339:                                     - ba[p][2]*radius;
340:                   /* 旋回中心を中心として車体を回転する */
341:                   co = radius / sqrt( radius*radius + run*run );
342:                   si = run / sqrt( radius*radius + run*run );
```

▶ 8月号，50ページの津村さんもお勧めの「ぷよぷよラムネ」ですが，ぷよにハンコがついたニューバージョンが発売されたので，前バージョンにハマッた私も再びハメられています。「赤ずきんチャチャキャンディー」といい○バヤ，侮りがたし。木村　奨(22)兵庫県

SIDE A

```
343:                    /* 位置ベクトルは旋回中心からの相対位置ベクトルを回転する */
344:                    b[1-p][0] = centerx + radius*co*ba[p][0] + radius*si*bc[p][0];
345:                    b[1-p][1] = centery + radius*co*ba[p][1] + radius*si*bc[p][1];
346:                    b[1-p][2] = centerz + radius*co*ba[p][2] + radius*si*bc[p][2];
347:                    /* 基底ベクトルはそのものを回転する */
348:                    /* bb(β軸)は旋回運動では変化しない */
349:                    ba[1-p][0] =  co*ba[p][0] + si*bc[p][0];
350:                    bc[1-p][0] = -si*ba[p][0] + co*bc[p][0];
351:
352:                    ba[1-p][1] =  co*ba[p][1] + si*bc[p][1];
353:                    bc[1-p][1] = -si*ba[p][1] + co*bc[p][1];
354:
355:                    ba[1-p][2] =  co*ba[p][2] + si*bc[p][2];
356:                    bc[1-p][2] = -si*ba[p][2] + co*bc[p][2];
357:
358:                    b[1-p][0] += bc[1-p][0]*testcar.wheelbase/2;
359:                    b[1-p][1] += bc[1-p][1]*testcar.wheelbase/2;
360:                    b[1-p][2] += bc[1-p][2]*testcar.wheelbase/2;
361:
362:                    phi = run/radius;
363:
364:                    wd1 = (radius - testcar.fshaft/2);
365:                    wd2 = sqrt( wd1*wd1 + (testcar.wheelbase)*(testcar.wheelbase) );
366:                    phi_fl += wd2*phi/testcar.fradius;
367:
368:                    wd1 = (radius + testcar.fshaft/2);
369:                    wd2 = sqrt( wd1*wd1 + (testcar.wheelbase)*(testcar.wheelbase) );
370:                    phi_fr += wd2*phi/testcar.fradius;
371:
372:                    phi_rl += (radius - testcar.rshaft/2)*phi/testcar.rradius;
373:                    phi_rr += (radius + testcar.rshaft/2)*phi/testcar.rradius;
374:            } else if ( x > 128 ) { /* 右折 */
375:                    roll = ROLL;                    /* にせ */
376:                    radius = testcar.wheelbase / tan( -theta );
377:                    centerx = b[p][0] - bc[p][0]*testcar.wheelbase/2
378:                                    + ba[p][0] * radius;
379:                    centery = b[p][1] - bc[p][1]*testcar.wheelbase/2
380:                                    + ba[p][1] * radius;
381:                    centerz = b[p][2] - bc[p][2]*testcar.wheelbase/2
382:                                    + ba[p][2] * radius;
383:                    co = radius / sqrt( radius*radius + run*run );
384:                    si = run / sqrt( radius*radius + run*run );
385:                    b[1-p][0] = centerx - radius*co*ba[p][0] + radius*si*bc[p][0];
386:                    b[1-p][1] = centery - radius*co*ba[p][1] + radius*si*bc[p][1];
387:                    b[1-p][2] = centerz - radius*co*ba[p][2] + radius*si*bc[p][2];
388:
389:                    ba[1-p][0] = co*ba[p][0] - si*bc[p][0];
390:                    bc[1-p][0] = si*ba[p][0] + co*bc[p][0];
391:
392:                    ba[1-p][1] = co*ba[p][1] - si*bc[p][1];
393:                    bc[1-p][1] = si*ba[p][1] + co*bc[p][1];
394:
395:                    ba[1-p][2] = co*ba[p][2] - si*bc[p][2];
396:                    bc[1-p][2] = si*ba[p][2] + co*bc[p][2];
397:
398:                    b[1-p][0] += bc[1-p][0]*testcar.wheelbase/2;
399:                    b[1-p][1] += bc[1-p][1]*testcar.wheelbase/2;
400:                    b[1-p][2] += bc[1-p][2]*testcar.wheelbase/2;
401:
402:                    phi = run/radius;
403:
404:                    wd1 = (radius + testcar.fshaft/2);
405:                    wd2 = sqrt( wd1*wd1 + (testcar.wheelbase)*(testcar.wheelbase) );
406:                    phi_fl += wd2*phi/testcar.fradius;
407:
408:                    wd1 = (radius - testcar.fshaft/2);
409:                    wd2 = sqrt( wd1*wd1 + (testcar.wheelbase)*(testcar.wheelbase) );
410:                    phi_fr += wd2*phi/testcar.fradius;
411:
412:                    phi_rl += (radius + testcar.rshaft/2)*phi/testcar.rradius;
413:                    phi_rr += (radius - testcar.rshaft/2)*phi/testcar.rradius;
414:            } else {        /* 直進 */
415:                    roll = 0.0;                     /* にせ */
416:                    b[1-p][0] = b[p][0] + bc[p][0] * run;
417:                    b[1-p][1] = b[p][1] + bc[p][1] * run;
418:                    b[1-p][2] = b[p][2] + bc[p][2] * run;
419:                    ba[1-p][0] = ba[p][0]; ba[1-p][1] = ba[p][1]; ba[1-p][2] = ba[p][2];
420:                    bb[1-p][0] = bb[p][0]; bb[1-p][1] = bb[p][1]; bb[1-p][2] = bb[p][2];
421:                    bc[1-p][0] = bc[p][0]; bc[1-p][1] = bc[p][1]; bc[1-p][2] = bc[p][2];
422:
423:                    wd1 = run/testcar.fradius;
424:                    phi_fl += wd1;
425:                    phi_fr += wd1;
426:                    wd1 = run/testcar.rradius;
427:                    phi_rl += wd1;
428:                    phi_rr += wd1;
429:            }
430:    }
431:    return;
432: }
433:
434: void      checkcollision()
435: {
436:    int     i, ry, rd;
437:    double  x, y, z, l;
438:
439:    /* 検索起点 */
440:    x = b[1-p][0] - (testcar.height*0.5)*bb[1-p][0];
441:    y = b[1-p][1] - (testcar.height*0.5)*bb[1-p][1];
442:    z = b[1-p][2] - (testcar.height*0.5)*bb[1-p][2];
443:    i = checks2( &ry, &rd, collide_checkinfolist, (int)x, (int)y, (int)z );
444:    if ( i == -1 ) return;
445:    /* 壁にぶつかったら前の状態に戻す */
446:    if ( collide_polygonlist->polygon[i].palet == TP_BACKGROUND ) {
447:             b[1-p][0] =  b[p][0];  b[1-p][1] =  b[p][1];  b[1-p][2] =  b[p][2];
448:            ba[1-p][0] = ba[p][0]; ba[1-p][1] = ba[p][1]; ba[1-p][2] = ba[p][2];
449:            bb[1-p][0] = bb[p][0]; bb[1-p][1] = bb[p][1]; bb[1-p][2] = bb[p][2];
450:            bc[1-p][0] = bc[p][0]; bc[1-p][1] = bc[p][1]; bc[1-p][2] = bc[p][2];
451:            return;
452:    }
453:    /* 新しい位置(y座標のみ変わる) */
454:    b[1-p][1] = (double)ry;
455:    /* 面法線を求める(checklib本体に組み込むのがスジか?) */
456:    x = -(double)collide_checkinfolist->ci[i].a;
457:    y = -(double)collide_checkinfolist->ci[i].b;
458:    z = -(double)collide_checkinfolist->ci[i].c;
459:    l = sqrt( x*x + y*y + z*z );
460:    x /= l;
461:    y /= l;
462:    z /= l;
463:    /* 新しいβ軸(面法線) */
464:    bb[1-p][0] = x;
465:    bb[1-p][1] = y;
466:    bb[1-p][2] = z;
467:    /* 新しいγ軸(補正前のγ軸から新β軸成分を取り除いて正規化したもの) */
468:    l = x*bc[1-p][0] + y*bc[1-p][1] + z*bc[1-p][2];
469:    x = bc[1-p][0] - l*x;
470:    y = bc[1-p][1] - l*y;
471:    z = bc[1-p][2] - l*z;
472:    l = sqrt( x*x + y*y + z*z );
473:    bc[1-p][0] = x/l;
474:    bc[1-p][1] = y/l;
475:    bc[1-p][2] = z/l;
476:    /* 新しいα軸(新β軸と新γ軸の外積) */
477:    ba[1-p][0] = bb[1-p][1]*bc[1-p][2] - bb[1-p][2]*bc[1-p][1];
478:    ba[1-p][1] = bb[1-p][2]*bc[1-p][0] - bb[1-p][0]*bc[1-p][2];
479:    ba[1-p][2] = bb[1-p][0]*bc[1-p][1] - bb[1-p][1]*bc[1-p][0];
480:    return;
481: }
482:
483: void      displayObject( SLPOINTLIST *pointlist, SLPOLYGONLIST *polygonlist, int mode )
484: {
485:    /* 呼びだし時に値を入れておく変数一覧 */
486:    /* MATRIX3 m1: ローカル座標系上の変換行列(タイヤなどキャラクタ内部のローカルな動き) */
487:    /* VECTOR3 v1: ローカル座標系上のオフセット(タイヤの取り付け位置など) */
488:    /* VECTOR3 v,va,vb,vc: 視線の座標系 */
489:    /* VECTOR3 b,ba,bb,bc: 車の基底座標系 */
490:
491:    invertmat2( m1b, m1 );
492:
493:    m0[0][0] = ba[1-p][0];
494:    m0[0][1] = ba[1-p][1];
495:    m0[0][2] = ba[1-p][2];
496:    m0[1][0] = bb[1-p][0];
497:    m0[1][1] = bb[1-p][1];
498:    m0[1][2] = bb[1-p][2];
499:    m0[2][0] = bc[1-p][0];
500:    m0[2][1] = bc[1-p][1];
501:    m0[2][2] = bc[1-p][2];
502:    invertmat2( m0b, m0 );
503:
504:    m2[0][0] = va[0];
505:    m2[0][1] = va[1];
506:    m2[0][2] = va[2];
507:    m2[1][0] = vb[0];
508:    m2[1][1] = vb[1];
509:    m2[1][2] = vb[2];
510:    m2[2][0] = vc[0];
511:    m2[2][1] = vc[1];
512:    m2[2][2] = vc[2];
513:
514:    multmat2( m3, m0b, m1b );
515:
516:    multmat2( m4, m2, m3 );
517:
518:    mat.r[0][0] = (int)(m4[0][0]*0x3FFF);
519:    mat.r[0][1] = (int)(m4[0][1]*0x3FFF);
520:    mat.r[0][2] = (int)(m4[0][2]*0x3FFF);
521:    mat.r[1][0] = (int)(m4[1][0]*0x3FFF);
522:    mat.r[1][1] = (int)(m4[1][1]*0x3FFF);
523:    mat.r[1][2] = (int)(m4[1][2]*0x3FFF);
524:    mat.r[2][0] = (int)(m4[2][0]*0x3FFF);
525:    mat.r[2][1] = (int)(m4[2][1]*0x3FFF);
526:    mat.r[2][2] = (int)(m4[2][2]*0x3FFF);
527:
528:    /* 位置 */
529:    w[0] =  b[1-p][0] + ba[1-p][0]*v1[0] + bb[1-p][0]*v1[1] + bc[1-p][0]*v1[2];
530:    w[1] =  b[1-p][1] + ba[1-p][1]*v1[0] + bb[1-p][1]*v1[1] + bc[1-p][1]*v1[2];
531:    w[2] =  b[1-p][2] + ba[1-p][2]*v1[0] + bb[1-p][2]*v1[1] + bc[1-p][2]*v1[2];
532:    v2[0] = w[0] - v[0];
533:    v2[1] = w[1] - v[1];
534:    v2[2] = w[2] - v[2];
535:    applymat2( v3, m2, v2 );
536:
637:    parameter.x = (int)v3[0];
538:    parameter.y = (int)v3[1];
539:    parameter.z = (int)v3[2];
540:    if ( mode == 0 ) {      /* 通常の描画 */
541:            TranslateByMatrixT( &parameter, work, pointlist, &mat );
542:            DisplayPolygonListT( polygonlist, work );
543:    } else {        /* 影用のラスタ抜き描画 */
544:            SetWindowSizeTR( 256, 256 );    /* ラスタ抜き用の環境構築 */
545:            SetWindowCenterTR( 128, 128 );
546:            if ( time%2 == 0 ) {
547:                    SetWritePlaneTR( (unsigned short *)0xE00020 );
548:                    SetClearPlaneTR( (unsigned short *)0xE00020 );
549:            } else {
550:                    SetWritePlaneTR( (unsigned short *)0xE00000 );
551:                    SetClearPlaneTR( (unsigned short *)0xE00000 );
552:            }
553:            TranslateByMatrixT( &parameter, work, pointlist, &mat );
554:            DisplayPolygonListTR( polygonlist, work );
555:            SetWindowSizeT( 256, 256 );     /* 環境を戻す */
556:            SetWindowCenterT( 128, 128 );
557:            if ( time%2 == 0 ) {
558:                    SetWritePlaneT( (unsigned short *)0xE00020 );
559:                    SetClearPlaneT( (unsigned short *)0xE00020 );
560:            } else {
561:                    SetWritePlaneT( (unsigned short *)0xE00000 );
562:                    SetClearPlaneT( (unsigned short *)0xE00000 );
563:            }
564:    }
565:
566:    return;
567: }
568:
569: void      display()
570: {
571:    double  ct, st, cp, sp;
572:    int     i;
573:
574:    if ( time%2 == 0 ) {
575:            SetWritePlaneT( (unsigned short *)0xE00020 );
576:            SetClearPlaneT( (unsigned short *)0xE00020 );
577:    } else {
578:            SetWritePlaneT( (unsigned short *)0xE00000 );
579:            SetClearPlaneT( (unsigned short *)0xE00000 );
580:    }
581:    ClearT();
582:
583:    /* 視点/視線 */
584:    if ( viewmode == 0 ) {
585:            /* 位置 */
586:            v[0] = b[1-p][0] - (testcar.height*20.0)*bb[1-p][0];
587:            v[1] = b[1-p][1] - (testcar.height*20.0)*bb[1-p][1];
588:            v[2] = b[1-p][2] - (testcar.height*20.0)*bb[1-p][2];
589:
590:            va[0] =  ba[1-p][0]; va[1] =  ba[1-p][1]; va[2] =  ba[1-p][2];  /* α軸 */
591:            vb[0] = -bc[1-p][0]; vb[1] = -bc[1-p][1]; vb[2] = -bc[1-p][2];  /* β軸 */
592:            vc[0] =  bb[1-p][0]; vc[1] =  bb[1-p][1]; vc[2] =  bb[1-p][2];  /* γ軸 */
```

▶次は683円ですか？　　竹内　周作(21)大阪府

```
593:    } else {
594:            vphi = 5.0*M_PI/180;;
595:            vlength = testcar.length*2.0;
596:            vheight = testcar.height*0.5;
597:            switch ( viewdirection ) {
598:            case 1: vtheta =   35.0*M_PI/180; break;
599:            case 2: vtheta =    0.0*M_PI/180; break;
600:            case 3: vtheta =  -35.0*M_PI/180; break;
601:            case 4: vtheta =   90.0*M_PI/180; break;
602:            case 6: vtheta =  -90.0*M_PI/180; break;
603:            case 7: vtheta =  145.0*M_PI/180; break;
604:            case 8: vtheta =  180.0*M_PI/180; break;
605:            case 9: vtheta = -145.0*M_PI/180; break;
606:            case 5: vtheta =  180.0*M_PI/180;
607:                    vphi   =   15.0*M_PI/180;
608:                    vheight = testcar.height*1.0;
609:                    vlength = testcar.length*(-0.1);
610:                    break;
611:            }
612:            m0[0][0] = ba[1-p][0];
613:            m0[0][1] = ba[1-p][1];
614:            m0[0][2] = ba[1-p][2];
615:            m0[1][0] = bb[1-p][0];
616:            m0[1][1] = bb[1-p][1];
617:            m0[1][2] = bb[1-p][2];
618:            m0[2][0] = bc[1-p][0];
619:            m0[2][1] = bc[1-p][1];
620:            m0[2][2] = bc[1-p][2];
621:            invertmat2( m0b, m0 );
622:
623:            vtheta = -vtheta+M_PI;
624:            vphi = -vphi;
625:            ct = cos( vtheta ); st = sin( vtheta );
626:            m2[0][0] =  ct; m2[0][1] = 0.0; m2[0][2] =  st;
627:            m2[1][0] = 0.0; m2[1][1] = 1.0; m2[1][2] = 0.0;
628:            m2[2][0] = -st; m2[2][1] = 0.0; m2[2][2] =  ct;
629:            cp = cos( vphi );   sp = sin( vphi );
630:            m3[0][0] = 1.0; m3[0][1] = 0.0; m3[0][2] = 0.0;
631:            m3[1][0] = 0.0; m3[1][1] =  cp; m3[1][2] = -sp;
632:            m3[2][0] = 0.0; m3[2][1] =  sp; m3[2][2] =  cp;
633:
634:            multmat2( m1, m2, m3 );
635:            invertmat2( m1b, m1 );
636:            invertmat2( m0b, m0 );
637:            multmat2( m4, m0b, m1 );
638:
639:            va[0] = m4[0][0]; va[1] = m4[1][0]; va[2] = m4[2][0];
640:            vb[0] = m4[0][1]; vb[1] = m4[1][1]; vb[2] = m4[2][1];
641:            vc[0] = m4[0][2]; vc[1] = m4[1][2]; vc[2] = m4[2][2];
642:
643:            v[0] = b[1-p][0] - vlength*vc[0] - vheight*bb[1-p][0];
644:            v[1] = b[1-p][1] - vlength*vc[1] - vheight*bb[1-p][1];
645:            v[2] = b[1-p][2] - vlength*vc[2] - vheight*bb[1-p][2];
646:    }
647:    /* 地面の表示 */
648:    mat.r[0][0] = (int)(va[0]*0x3FFF);
649:    mat.r[0][1] = (int)(va[1]*0x3FFF);
650:    mat.r[0][2] = (int)(va[2]*0x3FFF);
651:    mat.r[1][0] = (int)(vb[0]*0x3FFF);
652:    mat.r[1][1] = (int)(vb[1]*0x3FFF);
653:    mat.r[1][2] = (int)(vb[2]*0x3FFF);
654:    mat.r[2][0] = (int)(vc[0]*0x3FFF);
655:    mat.r[2][1] = (int)(vc[1]*0x3FFF);
656:    mat.r[2][2] = (int)(vc[2]*0x3FFF);
657:
658:    parameter.x = -(int)(v[0]);
659:    parameter.y = -(int)(v[1]);
660:    parameter.z = -(int)(v[2]);
661:    parameter.alpha = 8;
662:    parameter.beta = 8;
663:    parameter.palettype = SLPALETTYPE_NORMAL;
664:    GetRayViewT( &lalpha, &lbeta, &ldata, 8, 8 );
665:    mat.lalpha = lalpha;
666:    mat.lbeta = lbeta;
667:    mat.ldata = ldata;
668:    TranslateByMatrixViewT( &parameter, work, field_pointlist, &mat );
669:    DisplayPolygonListT( field_polygonlist, work );
670:
671:    /* パイロン表示制御 */
672:    for ( i = 0; i < npylon; i++ ) {
673:            v1[0] = b[1-p][0] - v[0];
674:            v1[1] = b[1-p][1] - v[1];
675:            v1[2] = b[1-p][2] - v[2];
676:            v2[0] = pylon[i][0] - b[1-p][0];
677:            v2[1] = pylon[i][1] - b[1-p][1];
678:            v2[2] = pylon[i][2] - b[1-p][2];
679:            pylondc[i] = ((v1[0]*v2[0]+v1[1]*v2[1]+v1[2]*v2[2])>0.0)?1:0;
680:    }
681:
682:    /* パイロン(車の向こう側) */
683:    for ( i = 0; i < npylon; i++ ) {
684:            if ( pylondc[i] == 0 ) continue;
685:            parameter.x = pylon[i][0]-(int)(v[0]);
686:            parameter.y = pylon[i][1]-(int)(v[1]);
687:            parameter.z = pylon[i][2]-(int)(v[2]);
688:            parameter.palettype = SLPALETTYPE_NORMAL;
689:            GetRayViewT( &lalpha, &lbeta, &ldata, 8, 8 );
690:            mat.lalpha = lalpha;
691:            mat.lbeta = lbeta;
692:            mat.ldata = ldata;
693:            TranslateByMatrixViewT( &parameter, work, &pylon_pointlist, &mat );
694:            DisplayPolygonListT( &pylon_polygonlist, work );
695:    }
696:
697: #define   displayFR       {¥
698:   st = sin( theta );¥
699:   ct = cos( theta );¥
700:   sp = sin( phi_fr );¥
701:   cp = cos( phi_fr );¥
702:   m2[0][0] =  ct; m2[0][1] = 0.0; m2[0][2] =  st;¥
703:   m2[1][0] = 0.0; m2[1][1] = 1.0; m2[1][2] = 0.0;¥
704:   m2[2][0] = -st; m2[2][1] = 0.0; m2[2][2] =  ct;¥
705:   m3[0][0] = 1.0; m3[0][1] = 0.0; m3[0][2] = 0.0;¥
706:   m3[1][0] = 0.0; m3[1][1] =  cp; m3[1][2] = -sp;¥
707:   m3[2][0] = 0.0; m3[2][1] =  sp; m3[2][2] =  cp;¥
708:   multmat2( m1, m3, m2 );¥
709:   v1[0] = +testcar.fshaft/2;¥
710:   v1[1] = -testcar.fradius;¥
711:   v1[2] = +testcar.wheelbase/2;¥
712:   displayObject( fwheel_pointlist, fwheel_polygonlist, 0 );}
713:
714: #define   displayFL       {¥
715:   st = sin( theta );¥
716:   ct = cos( theta );¥
717:   sp = sin( phi_fl );¥
718:   cp = cos( phi_fl );¥
719:   m2[0][0] =  ct; m2[0][1] = 0.0; m2[0][2] =  st;¥
720:   m2[1][0] = 0.0; m2[1][1] = 1.0; m2[1][2] = 0.0;¥
721:   m2[2][0] = -st; m2[2][1] = 0.0; m2[2][2] =  ct;¥
722:   m3[0][0] = 1.0; m3[0][1] = 0.0; m3[0][2] = 0.0;¥
723:   m3[1][0] = 0.0; m3[1][1] =  cp; m3[1][2] = -sp;¥
724:   m3[2][0] = 0.0; m3[2][1] =  sp; m3[2][2] =  cp;¥
725:   multmat2( m1, m3, m2 );¥
726:   v1[0] = -testcar.fshaft/2;¥
727:   v1[1] = -testcar.fradius;¥
728:   v1[2] = +testcar.wheelbase/2;¥
729:   displayObject( fwheel_pointlist, fwheel_polygonlist, 0 );}
730:
731: #define   displayRR       {¥
732:   sp = sin( phi_rr );¥
733:   cp = cos( phi_rr );¥
734:   m1[0][0] = 1.0; m1[0][1] = 0.0; m1[0][2] = 0.0;¥
735:   m1[1][0] = 0.0; m1[1][1] = cp;  m1[1][2] = -sp;¥
736:   m1[2][0] = 0.0; m1[2][1] = sp;  m1[2][2] =  cp;¥
737:   v1[0] = +testcar.fshaft/2;¥
738:   v1[1] = -testcar.fradius;¥
739:   v1[2] = -testcar.wheelbase/2;¥
740:   displayObject( fwheel_pointlist, fwheel_polygonlist, 0 );}
741:
742: #define   displayRL       {¥
743:   sp = sin( phi_rl );¥
744:   cp = cos( phi_rl );¥
745:   m1[0][0] = 1.0; m1[0][1] = 0.0; m1[0][2] = 0.0;¥
746:   m1[1][0] = 0.0; m1[1][1] = cp;  m1[1][2] = -sp;¥
747:   m1[2][0] = 0.0; m1[2][1] = sp;  m1[2][2] =  cp;¥
748:   v1[0] = -testcar.fshaft/2;¥
749:   v1[1] = -testcar.fradius;¥
750:   v1[2] = -testcar.wheelbase/2;¥
751:   displayObject( fwheel_pointlist, fwheel_polygonlist, 0 );}
752:
753: #define   displayBody     {¥
754:   sp = sin( pitch );¥
755:   cp = cos( pitch );¥
756:   st = sin( roll );¥
757:   ct = cos( roll );¥
758:   m3[0][0] = 1.0; m3[0][1] = 0.0; m3[0][2] = 0.0;¥
759:   m3[1][0] = 0.0; m3[1][1] =  cp; m3[1][2] = -sp;¥
760:   m3[2][0] = 0.0; m3[2][1] =  sp; m3[2][2] =  cp;¥
761:   m2[0][0] =  ct; m2[0][1] = -st; m2[0][2] = 0.0;¥
762:   m2[1][0] =  st; m2[1][1] =  ct; m2[1][2] = 0.0;¥
763:   m2[2][0] = 0.0; m2[2][1] = 0.0; m2[2][2] = 1.0;¥
764:   multmat2( m1, m2, m3 );¥
765:   v1[0] = 0.0;¥
766:   v1[1] = 0.0;¥
767:   v1[2] = 0.0;¥
768:   displayObject( &car_pointlist, &car_polygonlist, 0 );}
769:
770:   /* 影の表示 */
771:   m1[0][0] = 1.0; m1[0][1] = 0.0; m1[0][2] = 0.0;
772:   m1[1][0] = 0.0; m1[1][1] = 1.0; m1[1][2] = 0.0;
773:   m1[2][0] = 0.0; m1[2][1] = 0.0; m1[2][2] = 1.0;
774:   v1[0] = 0.0;
775:   v1[1] = 0.0;
776:   v1[2] = 0.0;
777:   displayObject( shadow_pointlist, shadow_polygonlist, 1 );
778:
779:   /* 車の表示 */
780:   switch ( (viewmode == 0)?5:viewdirection ) {
781:   case 1: displayFR;      /* 右前輪 */
782:           displayRR;      /* 右後輪 */
783:           displayBody;    /* 車体 */
784:           displayFL;      /* 左前輪 */
785:           displayRL;      /* 左後輪 */
786:           break;
787:   case 2:
788:   case 5: displayFR;      /* 右前輪 */
789:           displayFL;      /* 左前輪 */
790:           displayRR;      /* 右後輪 */
791:           displayRL;      /* 左後輪 */
792:           displayBody;    /* 車体 */
793:           break;
794:   case 3: displayFL;      /* 左前輪 */
795:           displayRL;      /* 左後輪 */
796:           displayBody;    /* 車体 */
797:           displayFR;      /* 右前輪 */
798:           displayRR;      /* 右後輪 */
799:           break;
800:   case 4: displayFR;      /* 右前輪 */
801:           displayRR;      /* 右後輪 */
802:           displayBody;    /* 車体 */
803:           displayFL;      /* 左前輪 */
804:           displayRL;      /* 左後輪 */
805:           break;
806:   case 6: displayFL;      /* 左前輪 */
807:           displayRL;      /* 左後輪 */
808:           displayBody;    /* 車体 */
809:           displayFR;      /* 右前輪 */
810:           displayRR;      /* 右後輪 */
811:           break;
812:   case 7: displayRR;      /* 右後輪 */
813:           displayFR;      /* 右前輪 */
814:           displayBody;    /* 車体 */
815:           displayRL;      /* 左後輪 */
816:           displayFL;      /* 左前輪 */
817:           break;
818:   case 8: displayRR;      /* 右後輪 */
819:           displayRL;      /* 左後輪 */
820:           displayFR;      /* 右前輪 */
821:           displayFL;      /* 左前輪 */
822:           displayBody;    /* 車体 */
823:           break;
824:   case 9: displayRL;      /* 左後輪 */
825:           displayFL;      /* 左前輪 */
826:           displayBody;    /* 車体 */
827:           displayRR;      /* 右後輪 */
828:           displayFR;      /* 右前輪 */
829:           break;
830:   }
831:
832:   /* パイロン(車のこちら側) */
833:   mat.r[0][0] = (int)(va[0]*0x3FFF);
834:   mat.r[0][1] = (int)(va[1]*0x3FFF);
835:   mat.r[0][2] = (int)(va[2]*0x3FFF);
836:   mat.r[1][0] = (int)(vb[0]*0x3FFF);
837:   mat.r[1][1] = (int)(vb[1]*0x3FFF);
838:   mat.r[1][2] = (int)(vb[2]*0x3FFF);
839:   mat.r[2][0] = (int)(vc[0]*0x3FFF);
840:   mat.r[2][1] = (int)(vc[1]*0x3FFF);
841:   mat.r[2][2] = (int)(vc[2]*0x3FFF);
842:   for ( i = 0; i < npylon; i++ ) {
```

▶通信を始めるようになって，早起きができるようになった。通信は健康的？
前田　桂史(20)青森県

```
843:            if ( pylondc[i] == 1 ) continue;
844:            parameter.x = pylon[i][0]-(int)(v[0]);
845:            parameter.y = pylon[i][1]-(int)(v[1]);
846:            parameter.z = pylon[i][2]-(int)(v[2]);
847:            parameter.palettype = SLPALETTYPE_NORMAL;
848:            GetRayViewT( &lalpha, &lbeta, &ldata, 8, 8 );
849:            mat.lalpha = lalpha;
850:            mat.lbeta = lbeta;
851:            mat.ldata = ldata;
852:            TranslateByMatrixViewT( &parameter, work, &pylon_pointlist, &mat );
853:            DisplayPolygonListT( &pylon_polygonlist, work );
854:    }
855:    return;
856: }
857:
858: int        main()
859: {
860:    double  fps;
861:    int     k;
862:
863:    create_field();
864:    collide_checkinfolist = createCheckInfoList( collide_polygonlist->n, collide_polygonlist
, collide_pointlist );
865:    generateCheckInfoList( collide_checkinfolist );
866:
867:    create_object( &testcar );
868:
869:    work = malloc( sizeof(SLTRANSWORK)*N_POINT );
870:
871:    CRTMOD( 14 );
872:    G_CLR_ON();
873:    B_CUROFF();
874:    MS_INIT();
875:    MS_CUROF();
876:    MS_LIMIT(0,0,255,255);
877:    MS_CURST(128,128);
878:    MS_CUROF();
879:    SKEY_MOD(0,0,0);
880:    sp = SUPER( 0 );
881:
882:    TPALET2(  0, TC_00 );
883:    TPALET2(  1, TC_01 );
884:    TPALET2(  2, TC_02 );
885:    TPALET2(  3, TC_03 );
886:    TPALET2(  4, TC_04 );
887:    TPALET2(  5, TC_05 );
888:    TPALET2(  6, TC_06 );
889:    TPALET2(  7, TC_07 );
890:    TPALET2(  8, TC_08 );
891:    TPALET2(  9, TC_09 );
892:    TPALET2( 10, TC_10 );
893:    TPALET2( 11, TC_11 );
894:    TPALET2( 12, TC_12 );
895:    TPALET2( 13, TC_13 );
896:    TPALET2( 14, TC_14 );
897:    TPALET2( 15, TC_15 );
898:
899:    SetWindowSizeT( 256, 256 );
900:    SetWindowCenterT( 128, 128 );
901:    SetReverse( 2 );
902:
903:    /* 車体 */
904:    p = 0;
905:    b[p][0]  = 0.0; b[p][1]  = 0.0; b[p][2]  = 0.0; /* 位置(ワールド座標) */
906:    ba[p][0] = 1.0; ba[p][1] = 0.0; ba[p][2] = 0.0; /* 基底ベクトル(α軸) */
907:    bb[p][0] = 0.0; bb[p][1] = 1.0; bb[p][2] = 0.0; /* 基底ベクトル(β軸) */
908:    bc[p][0] = 0.0; bc[p][1] = 0.0; bc[p][2] = 1.0; /* 基底ベクトル(γ軸) */
909:
910:    /* 経過時間を計測する */
911:    t1 = ONTIME();
912:
913:    /* ダブルバッファリングを開始する */
914:    setDoubleBufferMode( 0, 1 );    /* テキストVRAMのみ */
915:    startDoubleBuffer( 3 );         /* 最短3/60秒でページ切り替え */
916:
917:    /* パフォーマンスモニタをリセット */
918:    PMreset;
919:
920:    /* メインループ */
921:    for (;;) {
922:            /* マウス入力 */
923:            mscur = MS_CURGT();
924:            x = mscur/65536;
925:            y = mscur%65536;
926:            msdt = MS_GETDT()%65536;
927:            lb = msdt/256;
928:            rb = msdt%256;
929:            /* ESCキーで終了 */
930:            if ( BITSNS(0x00)&2 ) break;
931:            /* F1で視点モード切り替え */
932:            if ( BITSNS(0x0C)&8 ) {
933:                    while ( BITSNS(0x0C)&8 );
934:                    viewmode = 1 - viewmode;
935:            }
936:            /* テンキーで視線方向制御(サボってるので同時押ししたら作動しない) */
937:            if ( k = BITSNS(0x08) ) {
938:                    switch ( k ) {
939:                    case 0x01: viewdirection = 5; viewmode = 1; break;
940:                    case 0x02: viewdirection = 6; viewmode = 1; break;
941:                    case 0x08: viewdirection = 1; viewmode = 1; break;
942:                    case 0x10: viewdirection = 2; viewmode = 1; break;
943:                    case 0x20: viewdirection = 3; viewmode = 1; break;
944:                    }
945:                    while ( BITSNS(0x08) );
946:            } else if ( k = BITSNS(0x09) ) {
947:                    switch ( k ) {
948:                    case 0x08: viewdirection = 7; viewmode = 1; break;
949:                    case 0x10: viewdirection = 8; viewmode = 1; break;
950:                    case 0x20: viewdirection = 9; viewmode = 1; break;
951:                    case 0x80: viewdirection = 4; viewmode = 1; break;
952:                    }
953:                    while ( BITSNS(0x09) );
954:            }
955:            if ( BITSNS(0x06)&32 ) {          /* SPACEキーでポーズ */
956:                    while ( BITSNS(0x06)&32 );
957:                    t1 = ONTIME();  /* 時間を戻さないと車が吹っ飛ぶ */
958:            }
959:
960:            drive();
961:            checkcollision();
962:
963:            /* これから描画するページが裏ページになるのを待つ */
964:            waitDoubleBufferSync;
965:
966:            display();
967:
968:            /* パフォーマンスモニタのカウントは1周期に1回呼ぶ */
969:            PMcount;
970:
971:            /* 表示ページ切り替えを割り込みルーチンに許可する */
972:            toggleDoubleBufferApage();
973:
974:            time++;
975:
976:            /* phaseを反転する */
977:            p = 1 - p;
978:    }
979:
980:    /* ダブルバッファリング終了 */
981:    endDoubleBuffer();
982:
983:    /* パフォーマンスモニタから平均fps値を得る */
984:    fps = PMaverage;
985:
986:    /* テキスト画面をクリア */
987:    SetClearPlaneT( (unsigned short *)0xE00000 );
988:    SetWindowSizeT( 256, 256 );
989:    SetWindowCenterT( 128, 128 );
990:    ClearT();
991:    SetClearPlaneT( (unsigned short *)0xE00020 );
992:    SetWindowSizeT( 256, 256 );
993:    SetWindowCenterT( 128, 128 );
994:    ClearT();
995:
996:    /* ユーザモードに戻す */
997:    SUPER( sp );
998:
999:    /* 画面を戻す */
1000:   B_CURON();
1001:   CRTMOD( 16 );
1002:
1003:   /* キーバッファをクリアする */
1004:   KFLUSHIO( 0xFF );
1005:
1006:   /* ワークエリア/形状の領域を破棄する */
1007:   free( work );
1008:   destroy_object();
1009:   destroy_field();
1010:   destroyCheckInfoList( collide_checkinfolist );
1011:
1012:   /* 計算した平均fps値を表示して終了する */
1013:   printf( "¥n¥n¥n average = %2.2ffps¥n", fps );
1014:
1015:   return 0;
1016: }
```

リスト9 Makefile

```
 1: SHELL            = a:¥command.x
 2: SLASHLIBDIR      = ..¥lib3
 3: SLASHINCLUDEDIR  = ..¥include
 4: #CCOPTS           = -O -Wall -I$(SLASHINCLUDEDIR)
 5: CCOPTS           = -m68040 -O -Wall -I$(SLASHINCLUDEDIR)
 6: #CCOPTS           = -g -O -Wall
 7: #CCOPTS           = -fno-defer-pop -g -O -Wall
 8:
 9: %.o: %.s
10:    has -u -w $<
11:
12: %.o: %.c
13:    gcc $(CCOPTS) -c $<
14:
15: all: tdrive.x
16:
17: tdrive.x: tdrive.o tcourse.o pack.o slmath2.o car.o pylon.o
18:    gcc -o tdrive.x tdrive.o tcourse.o pack.o slmath2.o car.o pylon.o¥
19:            $(SLASHLIBDIR)¥stdcolor.a¥
20:            $(SLASHLIBDIR)¥_slashlib.a¥
21:            $(SLASHLIBDIR)¥tslashlib.a¥
22:            $(SLASHLIBDIR)¥slashlib.a
23:
24: tdrive.o: tdrive.c textcolor.h
25: tcourse.o: tcourse.c textcolor.h
26: pack.o: pack.c
27: car.o: car.s
28: pylon.o: pylon.s
29: slmath2.o: slmath2.c
30:
31: clean:
32:    if exist *.dat del -y *.dat
33:    if exist *.bak del -y *.bak
34:    if exist tdrive.o del tdrive.o
35:    if exist tcourse.o del tcourse.o
36:    if exist pack.o del pack.o
37:    if exist car.o del car.o
38:    if exist pylon.o del pylon.o
39:    if exist slmath2.o del slmath2.o
40:
41: distclean: clean
42:    if exist tdrive.x del tdrive.x
```

▶定価の値上げはX68000に因んでか？　　　堀井　努(20)静岡県

SIDE B

さらにテクスチャマッピングを考える

Yokouchi Takeshi 横内 威至

割り切りとアルゴリズムの成果によりテクスチャマッピングを動かすことができた
結局，満足できる結果とならなかったのが非常に悔やまれる
しかし，使い方しだいで面白いことができそうだ

友達が，あるサーキットのそばのゲームセンターで「デイトナUSA」を見ていた。プレイしていたのはそのサーキットで走っていたドライバーだったそうだ。その走りはさすがに狂っていたらしい。見ている側の動態視力ではとてもカバーできないレベルで操っていた，と彼はいっていた。当然，上級コースであっさりとコースレコードを塗り替えていた，ともつけ加えていた。

この話を聞いたとき，以前某番組で中嶋悟が某有名タレントに「ファイナルラップ」であっさりと負けていたのを思い出した。最近のゲームはいよいよシミュレータのレベルに近づきつつあるのだろうか。実際と挙動が違う，なんてのは過去の話。かなりのレベルで現実に近づいている。

いくらゲームとはいえ，それなりの理論でしっかりと制御されている以上，ゲームにおける車の挙動は，その車のクセと考えてもかまわないのではないだろうか。挙動が手足に伝わってこないからいまいち感覚がズレる，なんてのも戯言にすぎないのだ。ぜひとも，プロのドライバーに一度「バーチャレーシング」だとかで勝負してもらいたいものである。

部分的に成功

8月号で，難解ではあるがテクスチャマッピングの手抜き高速処理のための下準備を考えてみた。今月もさらに続けて考えていく。

結果からまずいってしまうが，やはりマッピングは重い処理ということを改めて認識させられた。ある程度は成功したのであるが，X68000上でリアルタイムシミュレーションに使用するのは，まず不可能。表示が大きくなると限りなく遅い。

まあ，16MHzでの目測だが，1秒で1枚程度まで描画速度が落ちてしまう。1ドットにつき掛け算2回は使用しているのでしかたないのかもしれない。

また，しっかりとしたテクスチャマッピングの画像がないので，本当に正しい画像かどうかは確認していない。その判断は写真を見たうえで各自に任せよう。私はこれで大丈夫と思っている。それっぽく投影されているし，小さければ気持ちよく動いてくれるし，テクスチャマッピングとして面白い処理になっているからこれはこれでOKとしよう。

そしてマッピングプログラムによって生成された画像が写真1,2,3である。元画像となるのは写真4の総天然色のテスト画像のようなやつ。写真1のほうは，十分テクスチャマッピングといえるのではないだろうか。写真2はかなり手前に平面をもってきている。このくらいになってしまうとボロが出ているのがわかるだろう。本来色の継ぎ目は直線として投影されなければならないのだが，計算の誤差，テーブルの貧弱さゆえにガタガタになってしまっている。実用レベルには達することができなかったのは明白。やはり切り捨てた要素が大きすぎたのだろうか。写真3はクリアせずに1回転させた画像。ちゃんと3Dの投影らしくなっているあたりは結構嬉しい。どうやら致命的なミスはなかったとひと安心である。

写真1 マッピング画像

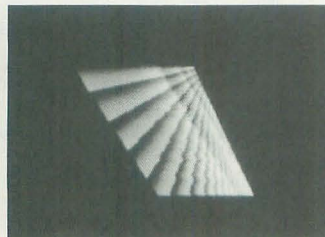



写真2 かなり手前にもってきた

写真3 クリアせずに1回転

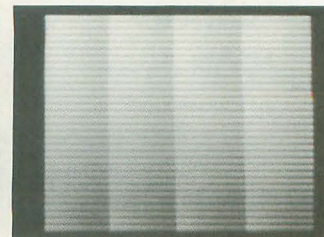



写真4 元画像

まあ，形はできたので，あとはうまいゴマカシと要領でどうにでもなるだろう，と楽観視しつつ，続きを考えよう。

総復習

まずは全体の処理をもう一度見直してみよう。それと同時に，しっかりとシステムに組み込むための処理も入れておく。具体的には三角形として描画させるために必要なことも考えておく。そして，このルーチンを実現するために用意したテーブルの構造を図1に示す。

1) 三角形の頂点を座標変換しておく
2) 3点の位置関係を調べ，左辺，右辺を決定
3) 投影される左辺，右辺のY方向1ドットごとのZ座標と，元画像のどこをその点に投影するかを決定する。左辺，右辺が対応する元画像上の点をa, bとする
4) 左辺と右辺の間を描画する。先ほど調べたZ座標によって，この1ライン上の各点のZ座標の比率を調べ，この比率によって先ほどのa, b点の間のドットを拾ってくる

以上が処理の簡単な流れである。先月考えた部分は3番目。1，2は以前ポリゴンの描画について考えた部分でそのまま通用する。では4番目だが，ここに移る前に先月説明した部分をしっかりと固めておくことにする。3番の処理についてなのだが，まずは直線を投影し，それと元絵の2点がどのように関係づけられるのかをしっかりと把握しなければならない。

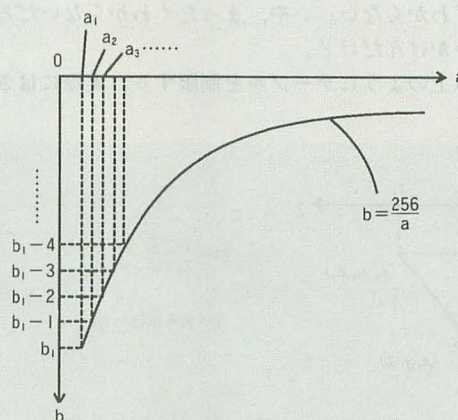
テーブルと傾き

まず，もう一度テーブルについて思い出そう。投影される2点を結ぶ直線上の各点のZ座標を求めるときに使うテーブルである。図2のような直線を投影したとき，Y座標とZ座標についての関係式は以下のようであった。

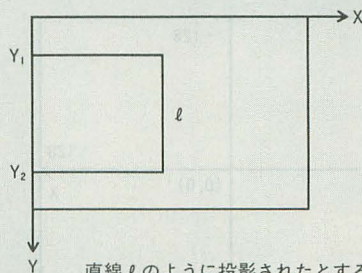
$$Y-256m=\frac{256(y_1-mz_1)}{z}$$

mの値は始点，終点のZ座標から求められるので，変数は当然Yとzの2つ。Yはスキャンライン順に1ずつ増える値なので，それにしたがってzが得られる。一定の条件のもとにテーブルの先頭を決め，順番にデータを拾って加工することでスキャンライン順のZ座標が得られるのである。(y_1-mz_1)を1として，Y−256mの範囲を適当に定めておいたときのzをテーブル化しておく。

そして，残った問題はテーブルをどこまでもつ必要があるか，である。これは前述のY−256mの範囲を定めることである。Yは投影されたY座標なので，座標系を図3のようにしておけば簡単に狭められる。消失点は一般的には画面中央としてかまわないので，Yの範囲は−128〜+127。これは，投影して描画される範囲だけに絞ってある。画面外に投影されるものはクリッピングするのだが，これについては図4で考えてみる。実はクリッピングは，範囲外の部分

図1 テーブルの構造

テーブルにはb_1, b_1-1, b_1-2, …に対応したa_1, a_2, a_3……が順番に入っている。

テーブルNo.	
1	a_1
2	a_2
3	a_3
4	
⋮	⋮
n	a_n

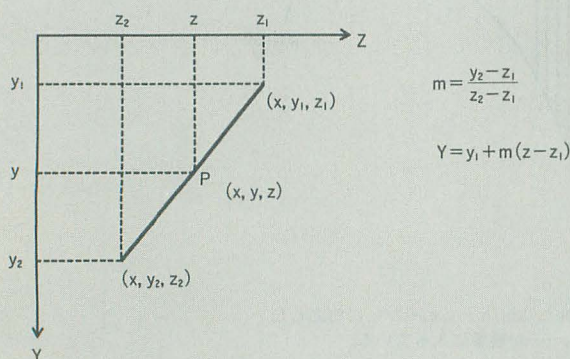
$Y-256m=\frac{256}{Z}(y_1-mz_1)$で，Yはスキャンラインの座標である。
Yが1ずつ増加するときのZの値を求めることが目的となる。

直線ℓのように投影されたとする。
$Y_1-256m=n$とする。
読み始めるテーブルの先頭はテーブル$n_0=n$から。
$Y=Y_1$ ：$Z=a_n\times(y_1-mz_1)$
$Y=Y_1+1$：$Z=a_{(n+1)}\times(y_1-mz_1)$
⋮
$Y=Y_2$ ：$Z=a(n+Y_2-Y_1)\times(y_1-mz_1)$

をいっさい無視するだけなのである。

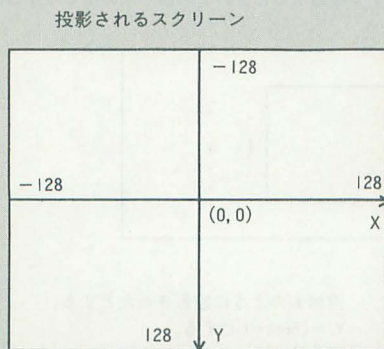
次にmについて考える。実際には−∞〜+∞となるのだが，一定範囲に絞ることでテーブルの量を現実的なものとする。これに関しては理論なしに直感で決めようと思う。m=±128で区切ってしまうことにした。mがこの範囲以外では画像に及ぼす影響はほとんど感じられない。X座標の差が128ドットでZ座標が1だけ違う状態と，Z座標が等しい状態をはっきりと見分けることができるだろうか。多分ほとんどわからない。いや，まったくわからないだろう。いいかげんだけど。

以上のようにテーブルを制限する。実際にはさらにmが負の状態も無視する。図5のようにmの符号は，テーブルから正方向か逆方向か，どちらを順番に拾うかを決定するだけとなる。

これでテーブルの範囲を有限に抑えることができた。クリッピングに関しても問題はなくなったのでテーブルのことはここまで。

図2 Y座標とZ座標の関係式

透視投影すると，

$$Y=256\frac{y}{z}$$
$$=256\frac{y_1+mz-mz_1}{z}$$
$$=256m+256\frac{y_1-mz_1}{z}$$
$$Y-256m=256\frac{y_1-mz_1}{z}$$

図3 投影する座標系

消失点を(0, 0)とする。

−128は画面内。しかし+128は画面の1ドット下。
よって，画面内でとりうる値は−128〜+127。

元絵のクリッピング

投影された直線のクリッピングなのだが，この直線のZ座標については前述のように問題ないことがわかっている。では元絵をどのようにクリッピングさせるかを考える。図6を見よう。単純にZ座標の比を使って元絵の2点を決め直すことで簡単に解決する。

心配なのは誤差。テーブルからデータを取る部分ですでにいいかげんであり，さらに適当な掛け算のせいでZ座標はそこそこ狂った値となってしまっている。実際に調べたところ，座標値で3ぐらいは平気でずれていた。これを単純にクリッピングにかけると，元絵の周辺までもが表示されていたり，あるいはその逆も起こりうる。ごまかしながら調整する必要があったのはこの部分。すべての誤差はこの部分で目立っていた。

しかし，この部分はアセンブラの内部事情にもからんでくるので，具体的な調整については気にしないことにした。結局，誤差がないもの（と決めつけた）としたうえで，アルゴリズムを追っていくことにする。

Z座標を計算する

総括してZ座標を得るまでを固める。ここはもう単純な作業だ。いままでのYとmからテーブルの先頭を決定し，そこからデータを拾ってくる。その値にy_1-mz_1を掛けてやればそのままZ座標となる。この座標値は直線全体との奥行きの比を意味し，それによって元絵の座標も決定される。ここは小数値で表すことにする。なぜ小数値を使うかというと，mの値が大きくなったときZ座標の整数部が以前計算した部分と同じになってしまうことが多いのだ。と，いってもこれは内部事情。いまいち理解しにくいかもしれない。

具体的にはmが無限大，あるいは+128以上のときに小数点以下の値が生きてくる。

まず，±∞のときは投影面と直線は平行。始点と終点の間もZ座標は一定。途中の各点は，元絵の始点と終点を等分割した点に比例して対応する。Z座標が同じだとすべて同じ部分を表示してしまうことになる。よって小数値を操作して適当に比例するよう

にしてごまかすのである。Z座標はこのあと左右の辺の間を描画するときに重要な値なので，影響のないように小数部で比を作っておくわけである。

左右の辺の間を描画する

ここが核心部分となる。ここまででスキャンライン順にZ座標と，その両端が元絵のどこに対応しているかは計算できた。あとここで必要なのはX座標である。これについては投影する座標を求めれば一発でわかる。左辺，右辺ともに直線を投影したものであり，投影後も当然直線となる。投影後の座標がわかっているので，ブレゼンハムのラインルーチンだとか普通の直線描画と同じ動作で，スキャンラインごとのX座標を求めることができる。

ではいよいよ，左右の座標，その点に対応した元画像の座標，そしてZ座標値から1スキャンライン分のテクスチャマッピングを行うことにする。といってもいままでと同じアルゴリズムで行うだけである。左右の辺を計算したときはZ座標を求めていたのだが，今度はスキャンライン上の各ドットごとの奥行きの比率だけが重要。元画像上の2点間から，この比率どおりの位置を各ドットに投影してやることが最終的な動作だ。

よって，Z座標を求めるアルゴリズムで奥行きの比率を求めることができる。ここまでわかれば基本的にはまったく問題ない。

図4 クリッピングについて

図のようにクリッピングすべき直線があるとする。
P_1からP_2まで1スキャンライン順にテーブルからデータを拾うのだが，初期値は図1の式より

Y−256m

にP_1の値を代入したものに対応する。
−256−256mがテーブルの頭だが，図のPからP_2までをテーブルから拾うことが目的。
つまり，どんな直線でもY座標が−128の点からだけを考えるわけであり，画面外の部分は無視できる。
テーブルでは$-128 \leqq Y < 128$の範囲だけを採用する。

図5 mの状態

図のように，

$$\ell_1 : m_1 = \frac{y_2 - y_1}{z_2 - z_1}$$

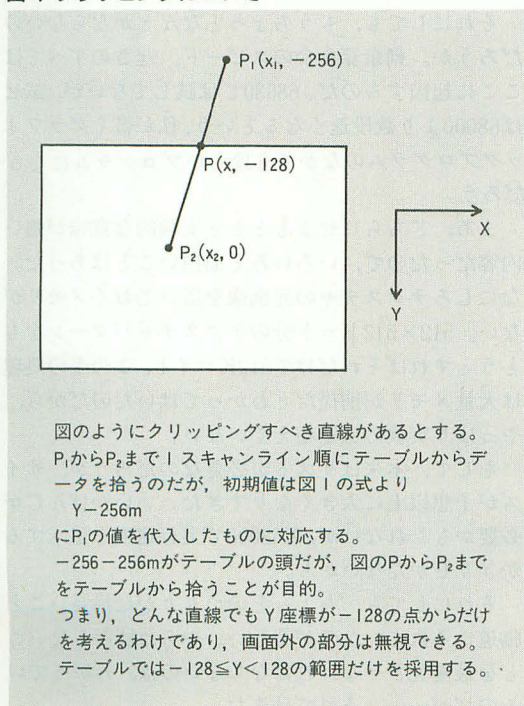
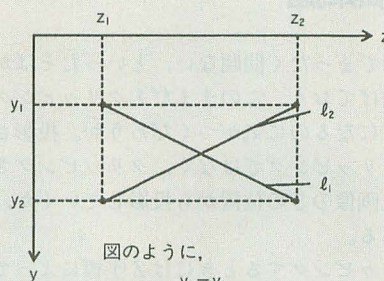
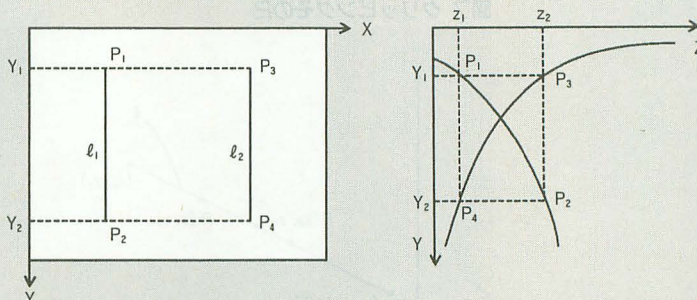
$$\ell_2 : m_2 = \frac{y_2 - y_1}{z_1 - z_2}$$

の2直線の傾きは$-m_1 = m_2$である。
投影すると下のように同じ直線となる。

Zについては右のようにテーブル化してあるグラフを反転させていることになる。符号が逆のときはテーブルから逆順にデータを拾えばよい。

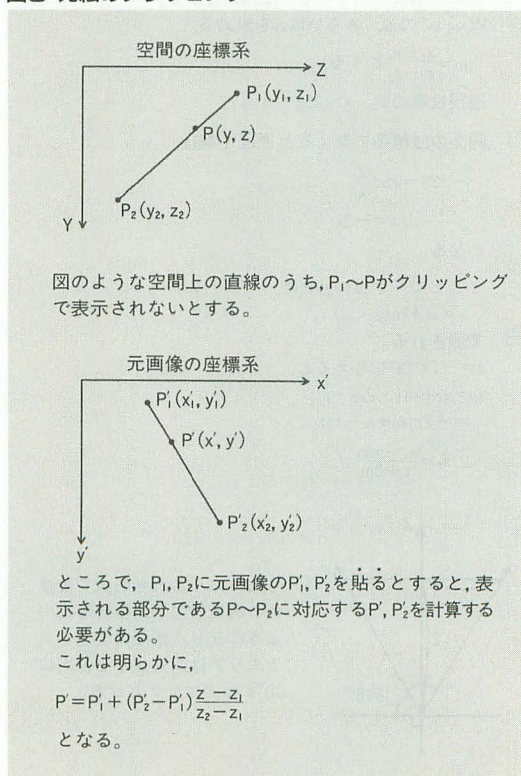
図6 元絵のクリッピング

図のような空間上の直線のうち，P_1〜Pがクリッピングで表示されないとする。

ところで，P_1, P_2に元画像のP'_1, P'_2を貼るとすると，表示される部分であるP〜P_2に対応するP', P'_2を計算する必要がある。
これは明らかに，

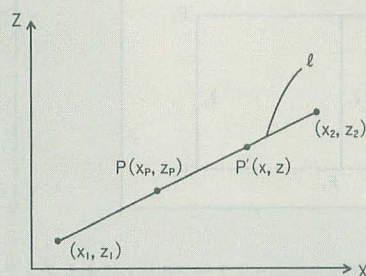
$$P' = P'_1 + (P'_2 - P'_1)\frac{z - z_1}{z_2 - z_1}$$

となる。

ハードコア3Dエクスタシー(第11回)

特殊な問題

すぐ手前でまったく問題ない，といったそばから問題点を挙げておく。このままだとクリッピングがマズイことになるのに気がつくだろうか。投影される座標のクリッピングではなく，クリッピングされた座標に元画像のどの位置から投影していくか，の段階でである。

辺をクリッピングするときにはZ座標によって計算していた。今度はZ座標は直線の傾きmを求めるときだけに使い，あとは奥行きの比率を求めただけなのでZ座標はわからない。そこでクリッピングされる端点だけのZ座標を求めることでうまくクリッピングして解決することにする(図7)。これで，投影された状態が空間のどの部分かを知ることもできる。

図7 クリッピングその2

図のような直線ℓがあり，透視投影すると必ずクリッピングされる画面外の部分があるとする。
画面左端にあたる点を$P(x_P, z_P)$とする。
$P(x_P, z_P)$のx_P，あるいはz_Pを求める。

$m=\frac{x_2-x_1}{z_2-z_1}$とする。

透視投影の式，$X=256\frac{z}{x}$から，

図2の座標系で考えると画面左端は，

$$-128=256\frac{z}{x}$$

$$\therefore z=-2x$$

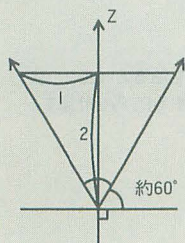
となる。

ところで，ℓ上の点P'は，

$x=x_1+m(z-z_1)$

で表される。

$z=-2x$でPを考えると，

$$x_P=x_1+m(-2x_P-z_1)$$
$$=-2mx_P+x_1-mz_1$$
$$\therefore x_P=\frac{x_1-mz_1}{1+2m}$$

ちなみに，これらの式で表される透視投影では視野は左のようになり，画面に表示されるエリアは消失点から左右に30°ずつのエリアとなる。

また，投影のクセがここではっきりとわかる。さらに図7のような関係からX＝−128，＋128でのZ座標が簡単に求められることもわかる。ここまでわかれば先ほどと同じようにクリッピングを行うことができる。

しかし，ここはもっとアルゴリズムを練らなければならない部分だ。理論は正しいのだが，アセンブラっぽく手抜きを行った今回のアルゴリズムをコーディングすると，誤差が結構大きい。予想ほどの誤差にはならなかったのだが，実際に見るとかなり心苦しいものがある。クリッピング部分はまだ考え直す必要がある。現段階ではまだバグも残っているし。

反省

素直に反省した。今回制作したものは半端なプログラムといえるだろう。スピードを追求した割にはリアルタイムで使うレベルのものではない。さらに画像が正確ではないのも問題だ。どうせ遅いならばしっかりと実数演算すべきなのだろうか。まあ，まともに実数演算をするよりは数倍以上速いことは確実なので，簡易VIEWERとしては十分かも。それと，まだ手をつけてないのはZ方向のクリッピング。これはマッピング描画の中核で行うのではなく，ルーチンに座標パラメータを与える部分で管理すべきものなので，今回は説明を省いてある。

それにしても，もうちょっとなんとかならないのだろうか。剰余算命令のスピード。遅さのすべてはここに起因するのだ。68030では試してないが，試せば68000より数段速くなるという，私が書くグラフィックプログラムのなかでも珍しいプログラムになるだろう。

まあ，どちらにせよもともと実験的な意味が強い内容だったので，いろいろと面白いことはあった。なにしろテクスチャの元画像を置いておくメモリがない。512×512ドット分のテクスチャパターンをもとうとすればそれだけで512Kバイト。この手の処理は大量メモリが前提だとわかってはいたのだから，やっぱり実験の域を超えていない。

そして，本来はリストが必要なのだろうが，サイズが予想以上に大きくなりすぎた。いいかげんCが必要かもしれないが，X68000で実行速度を追求するかぎりしかたないかも。

それにしても，ちょっとのつもりが結構長いこと脇道にそれていたのが苦しい。やらねばならないことを最優先させなければならないのは，わかっているのだが……。それではまた。

148部 怪しいZ80の使い方(テクニック編)

●Z80を使いこなしましょう

今月は，著者多忙のため「シューティングゲーム作成講座」をお休みして，MSX用S-OS "SWORD" 制作者としてお馴染みの筑紫氏による「怪しいZ80の使い方(テクニック編)」をお送りします。

雑誌などで紹介されたこともある広く知られた基本的なテクニックから，ヘタをすると自分自身でも理解不能となる危険を秘めているテクニックまで，幅広く取り上げています。

非力なCPUを補うため，1クロックでも少ない命令を組み合わせる努力。少ないメモリを犠牲にしてまでループを展開し，できるかぎりクロック削りに命をかける。これは，地道で苦痛を伴う作業です。

しかし，苦しんだぶん，きちんと見返りがあるはずです。あまり成果が出なかったとしても，それまで考え抜いた思考がきちんと経験として残ればいいのです。高速な命令の組み合わせ，新しいアルゴリズムを思いついたときには，非常に嬉しいものです(ときたま思い違いでエンバグするけど)。その喜びに向かって皆さんもガンガン高速化に励んでみてください。

●力は正義？

皆さんご承知のとおり，直交性に優れた68000に比べて，Z80はレジスタ，命令ともに貧弱に見える部分があります。

事実，高機能，高速なCPUにZ80が太刀打ちできるはずもありません。筆者も68000のアセンブラを初めて使ったときには，その便利さに唖然としたものです。まさに「なんでも許される」といった感じでしたからね(最初のうちは，ずいぶん無駄なコードをたくさん書いた記憶があります)。

しかし，いくら便利になったからといって，なにからなにまで簡単に実現できるかというとそうではありませんよね。やろうとしていることが，高度化していくのもありますが，やはりCPUパワーは有限です。

やはり，命令の性質を見抜く力をもつ人だけが，よりよいアルゴリズムを求め続けた人や，限界というものを超え続けることができるのでしょう。

頭の中で勝手に「限界だ」と思い込んであきらめるより，手と頭をフルに動かしましょう。

結局，CPU，ひいてはマシンを生かすも殺すもユーザーしだいです。古いものにこだわり続けろ，などというつもりはありませんが，自分が気にいったものなら，とことん使い込んでみましょう。

●オリジナルツールを紹介します

8月号で紹介した，L-os Angels。機能的には十分なものを備えていますが，このTHE SENTINELで掲載，もしくは配布するにあたってひとつ問題があります。それは，X1/turboのみのサポートだという点です。やはり，すべてのS-OSユーザーに使ってもらえる環境がなくてはいけません。残念ながら掲載は見送らせていただきますが，なにか動きがあれば，ぜひともTHE SENTINELまでご連絡ください。できるかぎり協力させていただきます。

ほかのサークルの皆さんも自分たちで制作した，オリジナルツールやパワーアップキットがあれば，いまいちど投稿してみてはいかがでしょうか。記事にできなくとも，このTHE SENTINELのコーナーをとおして，サークル以外の人々にも知ってもらうことはできます。配布費用については，シェアウェアとして適正な料金を徴収することでまかなえるはずです(いちばんの問題は，配布の手間かもしれませんが)。

ぜひ，プログラムと紹介文を添えて，THE SENTINELまでお送りください。

最後にS-OS "SWORD" MOOK化について。プログラムについては，フリーソフト化計画で集めたものを収録すればOK。問題のマニュアルを，有志を募ってユーザーの皆さんに作成していただき，最終的にOh!X編集部でMOOK化しよう，という方針で話がまとまりつつあります。いよいよ具体的に話が進みそうなMOOK化計画。いま一度，検討して話がもう少し具体的になったら，来月号で紹介します。

1994■インデックス

全機種共通
S-OS"SWORD"要

怪しいZ80の使い方(テクニック編)

Chikushi Takahiro
筑紫 高広

以前から紹介するといっていた，MSX用S-OS "SWORD" の制作者，筑紫氏によるZ80を使うためのテクニックを紹介します。マシン語バリバリのユーザーは参考にしてみてください。

8ビットCPUはあまりに遅いので，現在まで実行速度を補うべく，アルゴリズム，テクニックにより，高速化してきました。そこで，今回は僕がいままで培ってきたテクニックの数々を紹介していきます。

一応，Z80に対して使えるものばかりですが，ものによっては，CPUを超えて使用可能なものがあります。ぜひ参考にしてください。ここでは主に僕が某サークル誌に書いたテクニックを中心に解説します。

PUSH命令クリア

これは，某M氏が某サークル誌に書いておられた方法です。Z80のメモリクリアの最もオーソドックスなのは，LDIR命令による方法です。この方法だと，1バイトのクリアに約21クロックかかります。さらに1回転送するたびに，再度命令をフェッチしているようなので時間がかかってしまうようです（リフレッシュのためでしょうか)。R800でも，なぜか同じような感じです。

それに対してPUSHクリアは，1バイトを約5.5クロックでクリアすることができます。まず，SP（スタックポインタ）にクリアしたいエリアの最終＋1のアドレスをセットして，「PUSH HL」などの命令をたくさん並べるだけです（リスト1)。

リスト1 PUSHクリア

```
 1:     LD      (RETSP),SP
 2:     DI
 3:     LD      SP,最終+1
 4:     LD      HL,クリア・データ
 5:     PUSH    HL
 6:     PUSH    HL
 7:     PUSH    HL
 8:     PUSH    HL
 9:     PUSH    HL
10:     PUSH    HL
11:     PUSH    HL
12:     PUSH    HL
13:     PUSH    HL
14:     PUSH    HL
15:             :
16:     PUSH    HL
17:     EI
18:     DB      31H         ;LD SP,n
19: RETSP:
20:     DS      2
```

リスト2 POP-LD転送

```
 1:     LD      (RETSP),SP
 2:     DI
 3:     LD      SP,転送元先頭
 4:     POP     HL
 5:     LD      (転送先先頭),HL
 6:     POP     HL
 7:     LD      (転送先先頭+2),HL
 8:     POP     HL
 9:     LD      (転送先先頭+4),HL
10:             :
11:     POP     HL
12:     LD      (転送先最終-1),HL
13:     EI
14:     DB      31H         ;LD SP,n
15: RETSP:
16:     DS      2
```

もちろん，SPを使用するので，SPの値を保存して，割り込みを禁止する必要があります(このようにSPは，割り込みを禁止することで，本来のスタック以外にも使えるのです)。なお，メモリクリアは，2バイトセットで行われます。「DB 31H」は，ワークをプログラム中に埋め込んで高速化するという常套手段です。「EI」を先に実行しているのは，Z80では，割り込み許可が次の命令の実行後に行われるからです。つまり，可能な限り割り込みを許可した状態でプログラムを動かすようにするためです（割り込みの応答性を高めるため)。PUSHクリアを使えば，LDIR命令クリアより，約3.8倍速くクリアできます。

POP-LD転送およびPOP-OUT転送

以前どこかの，マイコン誌に，Z80の最高速の転送法は，POP-LD転送だと書いてありました。具体的な方法は書いてありませんでしたが，たぶん，リスト2のような方法だと思います。

リストを見ればわかるとおり，POP命令で，HLレジスタにデータを2バイト取り出し，LD命令で特定番地に転送しています。1バイトあたり約13クロックで転送できます。LDIR命令に比べて，約1.6倍のスピードです。なお，転送ルーチンは，転送するバイト数×2バイト程度が必要です。プログラムで，このルーチンを自動生成するようにするといいでしょう。

この転送法を応用して，POP-OUT転送も可能です。LD命令のところをOUT命令にするだけです。実際，MSX用S-OS "SWORD"では，文字列をPOP命令で取り込んで，OUT命令で表示しています。

MSX用S-OS"SWORD"の画面表示1

ついでですから，ここでPOP-OUT転送を使っている，MSX用S-OS"SWORD"の表示ルーチンについて解説します。

X1用S-OS"SWORD"の表示高速化パッケージ（未発表）に，深谷崇さんが開発した，HICONというものがありました(拡大回転縮小を使った，落下タマ転がし回転迷路ゲームや，F1ゲームやヘリコプターのゲ

ームなど，バリバリ動いていてすごいものです)。初期のルーチンをWINERで試したところ，LNPRNTルーチンを組み込んだ場合と，ほとんど同じ速さでスクロールしていました。

HICONの考え方は，従来のカーソル座標管理を，カーソルアドレスという方法に切り替えたものです。従来の方法では，x,y座標から，毎回カーソルアドレスを求めて表示していましたが，深谷さんのカーソルアドレス法は，カーソルアドレスそのものを座標管理に使用しているものです。たとえば，80桁表示の場合，右にカーソルを移動すると，カーソルアドレスを＋1，上にカーソルを移動すると，カーソルアドレスを－80するというものです。座標演算（通常，y×80＋x）を省略できるので，かなりの高速化ができるわけです。

テーブル演算

次に，座標計算について触れます。

Z80に限らず，よくテーブル演算という方法が取られます。つまり，アドレッシングを行うことで計算結果を求めるわけです。sinテーブルとかCRCテーブルとかよく聞くと思います。

要するに，掛け算九九表です。いくらカーソルアドレス法でも，任意の位置にカーソルを移動する場合，座標計算が必要なわけです。そこで，y座標×80またはy座標×40の，すべての組み合わせを計算しておくと，y座標に対応する数値をテーブルから読み込みx座標を加算することで，y座標×80＋x座標を計算することができます。

たとえば，ビットの左右を反転させる場合（8ビットで），256バイトのデータをあらかじめ計算しておき，テーブルアクセスします。テーブルは，通常，xx00$_H$というアドレスから生成します。

```
LD    H,MIRTBL/256
LD    L,A  ;HL←MIRTBL＋Acc
LD    A,(HL)  ;Acc←反転データ
```

まず，MIRTBLは，xx00$_H$から始まる，256バイトのビットの左右反転データです。最初に，Aレジスタには反転する前の値が入っています。LレジスタにAレジスタの値を入れると，HLレジスタが，Aレジスタの値に対応するテーブルのアドレスを示すことになり，無事反転された値を取り出すことができるのです。

MSX用S-OS“SWORD”の画面表示2

HICONの考え方を取り入れても（1文字表示ルーチンの構造は，HICONと基本的に同じです)，ハード，CPU(実質3MHzすら出ていない）のスピードの決定的な違いから，そのままでは，HICONに遠くおよびません。そこで，考え出したのが，文字列のブロック書き込みでした。POP-LD転送を応用したPOP-OUT転送です。この方法で，一時期，HICONの倍速弱になりましたが，現在では，双方の考え方を取り入れ，HICONは，さらに速くなっています。

MSX用S-OS“SWORD”の文字列表示ルーチンのコアの部分は，リスト3のようにしたと思います。これをループ展開すれば，究極のスピードになるはずです。1文字あたり39.5クロックですが（ノーウェイトなら，34.5クロックなので，ウェイトにより，12.7%速度が低下しています)，MSX2＋では，VDPアクセスに8μsec以下の間隔でアクセスしないようにウェイトが入るとのことなので，2ウェイト入り，1文字40.5クロックとなります。

しかし，実際は，1文字あたり48クロック程度となるようです。行末などの処理を考慮しても，ものすごくタイマ割り込みに，CPUスピードを食われているようです(十数%も!?)。

MSX用S-OS“SWORD”の特殊な座標管理

調子にのって，MSX用S-OS“SWORD”で行っている，特殊な座標管理について解説しましょう。通常のカーソル座標では，たとえば80桁の場合，x座標の左端が「0」，右端が「79」となっています。

しかし，MSX用S-OS “SWORD”では，右端が「FF$_H$」で，右へいくたびに，デクリメントしたようになっています。この方法を使うと，どんな利点があるかというと，ただひとつ，1文字表示ルーチンの行末チェックのみ，処理を高速化できるのです(ほかのすべての部分では，本来の値への「翻訳」が必要なので，スピードダウンしています)。

```
LD    HL,CSRXY
INC   (HL)
```

具体的には，この2つの命令でカーソル座標の更新と，行末の状態をZフラグに反映させることができるのです（ZF=1なら行末)。通常なら，以下のような感じになります。

```
LD    A,(CSRXY)
INC   A
LD    (CSRXY),A
CP    80
```

CSRXY＋1のアドレスに，筑紫式y座標が入っているとすると，通常の座標への変換は，次のようにします。ちなみに，80桁×25行の場合です。

```
        DS    21H       ;LD HL,n
CSRXY:  DS    2
        LD    DE,24*256+80
        ADD   HL,DE
```

DSの部分は，プログラムとワークを兼用した，常套手段です。加算する上位が24なのは，下位の加算で，必ず繰り上がりがあるため，1少ない値をセットしておくのです。

LDI転送

古籏一浩さんのSystem-7Bの，画面転送に使われていた方法です。LDIR転送では1バイト21クロックですが，LDI命令を並べる方法では，1バイトあたり16クロックで転送できます。転送するバイト数が多い場合，LDI命令をある程度並べてループするようにします。

割り算の四捨五入

通常の割り算ルーチンでは，小数点以下が捨てられます。これでは，誤差が最大＋1になります。四捨五入すれば，最大±0.5の誤差に収まり，精度が1ビット分高くなります。

リスト3 高速文字表示

```
1:          LD     A,1FH
2: MSGLP:   POP    HL
3:          CP     L
4:          JR     NC,MSGCT1
5:          OUT    (C),L
6:          CP     H
7:          JR     NC,MSGCT2
8:          OUT    (C),H
9:          DJNZ   MSGLP
```

つまり，商に0.5を足し，小数点以下を切り捨てるのです。プログラムで行うには，次のようにします（HL÷DE→HL)。

1)　被除数を２倍する。
2)　普通に，割り算ルーチン（四捨五入なし）を実行する。
3)　その結果（とりあえずの商）に１を足して，右シフトする。これは，0.5を足して，小数点以下を切り捨てる代わりです。

具体的には以下のようになります。

```
ADD   HL,HL    ;HL←HL×2
CALL  DIV66    ;HL←HL÷DE
INC   HL       ;HL←HL+1
SRL   H
RR    L        ;HL←int(HL÷2)
```

Z80常套手段

次にプログラミングするうえで，雑誌などでわりと紹介されたりしたことのある，基本的なテクニックを解説しましょう。

●Accのクリア

「XOR A」や「SUB A」がよく使われています。

●CYフラグのリセット

「OR A」を使います。

●定数の16ビット減算

SBC命令は遅く(M1が２回あるし)，CYフラグのリセットのために「OR A」などを事前に実行する必要がある場合があります。そこで，

```
LD   BC,-引く数
ADD  HL,BC
```

とすれば，速くなります。

●通常16ビットループ

ごく一般的な方法です。BCレジスタをカウンタに使います。BCレジスタをカウントダウンして，０の場合だけ，「OR C」の部分で，Aレジスタが０になることを利用します。

```
DEC  BC
LD   A,B
OR   C
JR   NZ,LOOP
```

●少し姑息な16ビットループ

上の方法より高速です。

```
ADD  HL,BC
JR   C,LOOP
```

Ｚフラグが変化しないので，CYフラグで判断します。CYフラグが立っている間ループします。HLレジスタにループする回数－1（1回余分にループするため），BCレジスタに－１（$FFFF_H$）をセットしておきます。

●もっと姑息な16ビットループ

さらに高速です。ループカウンタのBCの上位と下位を入れ換え，Ｂが０以外なら「INC　C」するという処理を加えて，次のようにします。

```
LOOP:   処理本体
DJNZ    LOOP
DEC     C
JR      NZ,LOOP
```

この方法が，たぶん，最高速の16ビットループ法です。

ループの挙動ですが，たとえば，200_H回だと次のようになります。

```
初回   256回
２回   256回
```

ところが，端数がある場合は，次のようになります（たとえば，201_H～$2FF_H$回)。

```
初回   端数（１～255）
２回   256回
３回   256回
```

このような場合，上位が「２」であるのに，端数の１回を余分に回らなくてはならないため，外側のループカウンタを１多くします。

この方法は，MSXのDISK-ROMで実際に使われていました。

アドレスのカウントアップ

アドレスのカウントアップは，通常「INC HL」などとしています。しかし，$xxFF_H$でない場合（256バイト境界）は，「INC L」を使っても，動作はほぼ同じ（フラグが変化する）で，３クロック速くなります。

たとえば，自作の網目モノクロハードコピープログラムEXHCのX1版では，パレットコードの取り出しに，この方法を使っています。640ドット(80桁)モードの場合は，横に80バイトずつデータが連続します。80という数字は「2^3×5」ですね。ということは，任意のラインを考えた場合，256バイト境界は，「2^3」ごとにやってくる可能性があるのです。逆にいえば，16バイトごとの最後のバイト以外は，「INC HL」などの代わりに，「INC L」が使えます。

フラグ

プログラムでは，２つの状態を識別するために，「フラグ」というものをワークエリアに確保します（レジスタのフラグと違い，ソフトウェア的に特定番地をフラグとして使います)。「フラグ」には，ビットごとに意味をもたせる場合と，１バイトをひとつの「フラグ」として使う場合があります。ビットごとの場合は，ワークエリアは小さくなりますが，ビットチェックのためにプログラム側が大きくなる，通常，ワークを節約するときのみ使います。

１バイトをひとつの「フラグ」として使う場合を説明します。僕は，FF_Hと０をフラグの値としています。通常は，０以外と０を使っていますが，このほうがいろいろと便利な部分があるのです。

「フラグ」をチェックするには，Ａレジスタに値を読み出したあと，「OR A」などでＺフラグに状態を反映させます。Ａレジスタの値が０の場合のみ，Ｚフラグが１になります。プログラムでは，次のようにします。

```
LD    A,(FLAG)
OR    A
```

CYフラグの状態をソフトウェアの「フラグ」に反映させるには，「SBC A,A」を実行します。この１命令で，CY＝0の場合はＡレジスタ＝00_H，CY＝1の場合はＡレジスタ＝FF_Hになります。

```
SBC    A,A
LD     (FLAG),A
```

「フラグ」が00_HとFF_Hのみの場合の便利な点ですが，たとえば「フラグ」が０ならＡレジスタに３，FF_HならＡレジスタに５を代入したい場合は，次のようにします。通常は，条件ジャンプで場合分けなどをしますが，以下の方法だと，その必要がないわけです。

```
LD   A,(FLAG)   ;00Hか FFH
AND  3          ;00Hか03H
ADD  A,2        ;02Hか05H
```

以上でZ80についてのテクニック集の紹介を終わります。来月はZ80の未定義命令をちょいちょいと紹介していく予定ですので，お楽しみに。

命令のクロック数は？

プログラムを最適化するうえで絶対必要になるものは，すべての命令表とその命令の実行速度です。すべてを覚える必要はありませんが，ある程度命令を把握しておかないとプログラムは組めませんからね。それにプログラムを組みたびにいちいち命令表を確認するのは，非常に面倒臭いものです。

さらにマシン語コートまで覚えれば，ダンプリストを直接いじり回したり，プログラムさえ組めます（一見ホラ話のように聞こえますが，実際にこういう人はいるのです)。

ここでは，すべての命令とまではいきませんが，代表的なもの，よく使われるものの命令クロックをずらずら書き並べていく予定です。今回は，転送命令，ブロック転送命令，ジャンプ命令について紹介します。

●転送命令

・8ビットイミディエイト値転送

```
LD    A,n
LD    B,n
LD    C,n
LD    D,n
LD    E,n
LD    H,n
LD    L,n
```

すべて7クロック

・8ビットレジスタ間転送命令

```
LD    A~L,A~L
```

すべて4クロック

・8ビット，メモリ←→レジスタ間転送

```
LD    A,(nn)
LD    (nn),A
```

以上，13クロック

```
LD    A,(BC)
LD    (BC),A
LD    A,(DE)
LD    (DE),A
LD    A~L,(HL)
LD    (HL),A~L
```

以上，7クロック

```
LD    A~L,(IX+d)
LD    A~L,(IY+d)
LD    (IX+d),A~L
LD    (IY+d),A~L
LD    (IX+d),n
LD    (IY+d),n
```

以上，19クロック

・16ビットイミディエイト値転送

```
LD    BC,Im
LD    DE,Im
LD    HL,Im
LD    SP,Im
```

以上，10クロック

```
LD    IX,Im
LD    IY,Im
```

以上，14クロック

・16ビット，メモリ←→レジスタ間転送

```
LD    HL,(Im)
LD    (Im),HL
```

以上，16クロック

```
LD    BC,(Im)
LD    DE,(Im)
LD    IX,(Im)
LD    IY,(Im)
LD    SP,(Im)
LD    (Im),BC
LD    (Im),DE
LD    (Im),IX
LD    (Im),IY
LD    (Im),SP
```

以上，20クロック

・その他

```
LD    A,I
LD    A,R
LD    I,A
LD    R,A
```

以上，9クロック

```
LD    SP,HL
```

以上，6クロック

```
LD    SP,IX
LD    SP,IY
```

以上，10クロック

●ブロック転送命令

```
LDD
LDI
```

以上，16クロック

```
LDDR
LDIR
```

1バイト転送につき21クロック。ただし，最終バイト転送のみ16クロック

●ジャンプ命令

・無条件ジャンプ

```
JP    (HL)
```

以上，4クロック

```
JP    (IX)
JP    (IY)
```

以上，8クロック

```
JP    Im
```

以上，10クロック

```
JR    e
```

以上，12クロック

・条件付ジャンプ

```
JP    NZ,Im
JP    Z,Im
JP    NC,Im
JP    C,Im
JP    PO,Im
JP    PE,Im
JP    P,Im
JP    M,Im
```

以上，10クロック

```
JR    NZ,e
JR    Z,e
JR    NC,e
JR    C,e
```

以上，条件成立時12クロック，不成立時7クロック

▶全機種共通システムインデックス◀

＊以下のアプリケーションは，基本システムであるS-OS"MACE"またはS-OS"SWORD"がないと動作しませんのでご注意ください。

第22部　magiFORTH TRACER
第23部　ディスクダンプ＆エディタ
第24部　"SWORD" 2000 QD
連載　対話で学ぶmagiFORTH
特別付録　PC-8801版S-OS "SWORD"
■86年7月号
第25部　FM音源ミュージックシステム
付録　FM音源ボードの製作
連載　計算力アップのmagiFORTH
特別付録　SMC-777版S-OS "SWORD"
■86年8月号
第26部　対局五目並べ
第27部　MZ-2500版S-OS "SWORD"
■86年9月号
第28部　FuzzyBASIC発表
連載　明日に向かってmagiFORTH
■86年10月号
第29部　ちょっと便利な拡張プログラム
第30部　ディスクモニタDREAM
第31部　FuzzyBASIC料理法<1>
■86年11月号
第32部　パズルゲームHOTTAN
第33部　MAZE in MAZE
連載　FuzzyBASIC料理法<2>
■86年12月号
第34部　CASL & COMET
連載　FuzzyBASIC料理法<3>

1987

■87年1月号
第35部　マシン語入力ツールMACINTO-C
連載　FuzzyBASIC料理法<4>
■87年2月号
第36部　アドベンチャーゲームMARMALADE
第37部　テキアベ作成ツールCONTEX
■87年3月号
第38部　魔法使いはアニメがお好き
第39部　アニメーションツールMAGE
付録　"SWORD" 再掲載とMAGICの標準化
■87年4月号
第40部　INVADER GAME
第41部　TANGERINE
■87年5月号
第42部　S-OS "SWORD" 変身セット
第43部　MZ-700用 "SWORD" をQD対応に
■87年6月号
インタラプト　コンパイラ物語
第44部　FuzzyBASICコンパイラ
第45部　エディタアセンブラZEDA-3
■87年7月号
第46部　STORY MASTER
■87年8月号
第47部　パズルゲーム碁石拾い
第48部　漢字出力パッケージJACKWRITE
特別付録　FM-7/77版S-OS "SWORD"
■87年9月号
第49部　リロケータブル逆アセンブラInside-R
特別付録　PC-8001/8801版S-OS "SWORD"
■87年10月号
第50部　tiny CORE WARS
第51部　FuzzyBASICコンパイラの拡張
第52部　X1turbo版S-OS "SWORD"
■87年11月号
序論　神話のなかのマイクロコンピュータ
付録　S-OSの仲間たち
第53部　もうひとつのFuzzyBASIC入門
第54部　ファイルアロケータ＆ローダ
インタラプト　S-OSこちら集中治療室
第55部　BACK GAMMON
■87年12月号
第56部　タートルグラフィックパッケージTURTLE
第57部　X1turbo版 "SWORD" アフターケア
ラインプリントルーチン
特別付録　PASOPIA7版S-OS "SWORD"

1988

■88年1月号
第58部　FuzzyBASICコンパイラ・奥村版
付録　石上版コンパイラ拡張部の修正
■88年2月号
第59部　シューティングゲームELFES
■88年3月号
第60部　構造型コンパイラ言語SLANG
■88年4月号
第61部　デバッギングツールTRADE
第62部　シミュレーションウォーゲームWALRUS
■88年5月号
第63部　シューティングゲームELFES II
第64部　地底最大の作戦
■88年6月号
第65部　構造化言語SLANG入門(1)
第66部　Lisp-85用NAMPAシミュレーション
■88年7月号
第67部　マルチウィンドウドライバMW-1
連載　構造化言語SLANG入門(2)
■88年8月号
第68部　マルチウィンドウエディタWINER
■88年9月号
第69部　超小型エディタTED-750
第70部　アフターケアWINERの拡張
■88年10月号
第71部　SLANG用ファイル入出力ライブラリ
第72部　シューティングゲームMANKAI
■88年11月号
第73部　シューティングゲームELFES IV
■88年12月号
第74部　ソースジェネレータSOURCERY

1989

■89年1月号
第75部　パズルゲームLAST ONE
第76部　ブロックゲームFLICK
■89年2月号
第77部　高速エディタアセンブラREDA
特別付録　X1版S-OS "SWORD" <再掲載>
■89年3月号
第78部　Z80用浮動小数点演算パッケージSOR
OBAN
■89年4月号
第79部　SLANG用実数演算ライブラリ
■89年5月号
第80部　ソースジェネレータRING
■89年6月号
第81部　超小型コンパイラTTC
■89年7月号
第82部　TTC用パズルゲームTICBAN
■89年8月号
第83部　CP/M用ファイルコンバータ
■89年9月号
第84部　生物進化シミュレーションBUGS
■89年10月号
第85部　小型インタプリタ言語TTI
■89年11月号
第86部　TTI用パズルゲームPUSH BON!
■89年12月号
第87部　SLANG用リダイレクションライブラリDIO.LIB

1990

■90年1月号
第88部　SLANG用ゲームWORM KUN
特別付録　再掲載SLANGコンパイラ
■90年2月号
第89部　超小型コンパイラTTC++
■90年3月号
第90部　超多機能アセンブラOHM-Z80
■90年4月号
第91部　ファジィコンピュータシミュレーションI-MY
■90年5月号
第92部　インタプリタ言語STACK
■90年6月号
第93部　リロケータブルフォーマットの取り決め
第94部　STACK用ゲームSQUASH!
第95部　X68000対応S-OS "SWORD"
特別付録　PC-286対応S-OS "SWORD"
■90年7月号
第96部　リロケータブルアセンブラWZD
■90年8月号
第97部　リンカWLK
■90年9月号
第98部　BILLIARDS
■90年10月号
第99部　ライブラリアンWLB
■90年11月号
第100部　タブコード対応エディタEDC-T
■90年12月号
第101部　STACKコンパイラ

1991

■91年1月号
第102部　ブロックアクションゲームCOLUMNS
■91年2月号
第103部　ダイスゲームKISMET
■91年3月号
第104部　アクションゲームMUD BALLIN'
■91年4月号
第105部　SLANG用カードゲームDOBON
■91年5月号
第106部　実数型コンパイラ言語REAL
■91年6月号
第107部　Small-C処理系の移植
■91年7月号
第108部　REALソースリスト編
■91年8月号
第109部　Small-Cライブラリの移植
■91年9月号
第110部　SLANG用NEWファイル出力ライブラリ
■91年10月号
第111部　Small-C活用講座（初級編）
■91年11月号
第112部　Small-C活用講座（応用編）
第113部　MORTAL
■91年12月号
第114部　Small-C SLANGコンパチ関数

1992

■92年1月号
第115部　LINER
■92年2月号
第116部　シミュレーションゲームPOLANYI
■92年3月号
第117部　カードゲームKLONDIKE
■92年4月号
第118部　オプティマイザO80実践Small-C講座(1)
■92年5月号
第119部　COMMAND.OBJ実践Small-C講座(2)
■92年6月号
第120部　COMMAND.OBJ2実践Small-C講座(3)
■92年7月号
第121部　関数リファレンス実践Small-C講座(4)
■92年8月号
第122部　ワイルドカード実践Small-C講座(5)
第123部　グラフィックライブラリ GRAPH.LIB
■92年9月号
第124部　O-EDIT&MODCNV
■92年10月号
第125部　SLENDER HUL実践Small-C講座(6)
■92年11月号
第126部　EDIT実践Small-C講座(7)
■92年12月号
第127部　MAKE実践Small-C講座(8)

1993

■93年1月号
第128部　EDC-Tの拡張
■93年2月号
第129部　BLACK JACK
■93年3月号
第130部　シューティングゲームコアシステム作成法(1)
■93年4月号
第131部　シューティングゲームコアシステム作成法(2)
■93年5月号
第132部　シューティングゲームコアシステム作成法(3)
■93年6月号
第133部　REVERSI
■93年7月号
特別付録　MSX用S-OS "SWORD"
■93年8月号
第134部　MACINTO-C再掲載
■93年9月号
第135部　7並べ
特別付録　SLANG再々掲載
■93年10月号
第136部　シューティングゲームコアシステム作成法(4)
■93年11月号
第137部　S-OSで学ぶZ80マシン語講座(1)
■93年12月号
第138部　エディタアセンブラREDA再掲載

こちらシステムX探偵事務所

FILE-XV

illustration : T.Takahashi

マルチボールと2階建て構造を目指す

Shibata Atsushi 柴田 淳

ひと月のお休みをいただいて，柴田氏が帰ってきました。ピンボールの作成はどうやら順調に進んでいるようです。今回はマルチボールを実現するためのボール同士の衝突や立体感を出すためのクリッピングについて考えていきます。

フィールド上にひとつずつ打ち出すボールを，複数に増やすことができる「マルチボール」。最近のピンボール台のマルチボールには，狭いレーンにボールを通してやるなどといった特定の条件を満たすものと，1ゲーム中に1回，ボタンを押すことで実現するものの2つのタイプがあるようである(どちらかといえば前者が主流)。いずれのタイプにしろ，フィールド上のボールの数が増えれば高得点ボーナスを得るチャンスは増える。10年以上前からとどまることをしらない得点のインフレーションと相まって，最近のピンボールでマルチボールというフィーチャーをもたないものはない，といっても過言ではない。

いままで，ボールの挙動をシミュレートするに当たっては「ボールと平面の反射」という，特殊化した状況ばかり扱ってきた。「特殊化した」というのは，ボールの進行方向と当たった壁の角度という，シミュレーションにおける重要な情報がどちらも明らかになっている，という意味である。さらに，衝突によって動きに影響を受けるのはボールの側だけなので，処理はいくぶん楽になる。

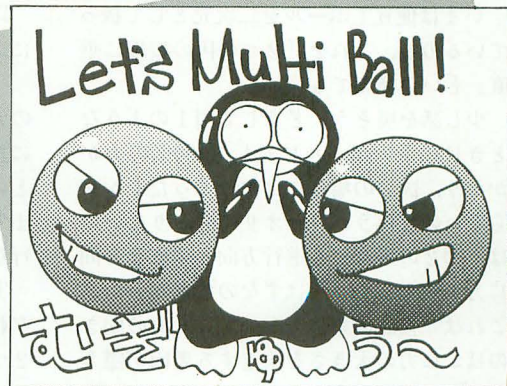
パソコン上のピンボールシミュレータにマルチボールを取り入れようとすれば，当然ボール同士の衝突，という状況を考慮しなければならない。壁とボールの衝突と違い，ボールとボールが衝突する場合はボールそれぞれが何らかの力を受け，進行方向を変える。つまり，考慮に入れるべき変数が倍になり，計算は繁雑になる。また，2つのボールの進行方向が同じでも，衝突時におけるお互いの位置関係によっては跳ね返り方も変わってくるだろう。このあたりの計算を，ボールの移動ベクトル，衝突した位置(ただし，衝突判定は当然必要)から得られる情報によってこなさなければならないのである。

ボール対ボールの衝突

さて，ボール同士の衝突という問題を，例によって図を見ながら検証するところから始めよう。とはいえ，いきなり問題の本質に迫るのではなく，まず状況を単純化してみる。

ボールとボールの衝突において(2つとも動いているとして)いちばん単純なのは，多分ボールの移動ベクトルが同一直線上に並び，2つのボールが正面衝突する場合だろう。ボール同士はだんだん距離を縮めていき，ぶつかって，やってきたのと正反対の方向にお互いを弾き，遠ざかっていく。この場合すべての動作は1本の直線上で起こるので，問題を一次元的な領域に押し込めることができる。つまり，力の及ぼされる方向に関しては考えずに，力の強さだけを考えればいいのだ。

図1のように，2つのボールが衝突する場合を想定しよう。とりあえずボールA(以下A)の受ける力に関してだけ考える。まず，衝突したことによってAの移動ベクトルと正反対の向きをもつ抗力$-\vec{A}$が働く。これにボールB(以下B)がAに及ぼす力，すなわちBの移動ベクトルがかかり，結局Aの受ける力は「$-\vec{A}+\vec{B}$」となる。これにもともとのAの移動ベクトルを足すと，$\vec{A}$と$-\vec{A}$が打ち消し合い，結局残るのは$\vec{B}$だけである。つまり，AはBの移動ベクトルをそのまま受け継ぐのだ。これと同じことがBにも起こる。2つのボールは衝突することにより移動ベクトルを交換し，互いに離れていくのである。

では，図2のように2つのボールの進行方向が平行で，位相がずれていて，お互いの最上部と最下部をかすめていく場合はどうだろうか。この場合，2つのボールは「衝突する」のではなくむしろ「すれ違う」のだから，お互いに力を及ぼし合わず，したがって移動方向に変化は現れない。この場合と，図1のような状況の「差」はなんだろうか。

どうやら，ボールが衝突して及ぼされる「力の方向」について考えてみる必要があるようだ。問題を簡単にするために，衝突する2つのボールの進行方向は互いに水平でなおかつ逆方向，ということにしよう。で，図3にあるようなもう少し日常的なボールの衝突の際，Aにかかる力の方向について考察するわけだが，ここで力学のセオリーを引き合いに出そう。

「2つの物体に働く力の方向は，接点に垂

図1 直線上の衝突の場合

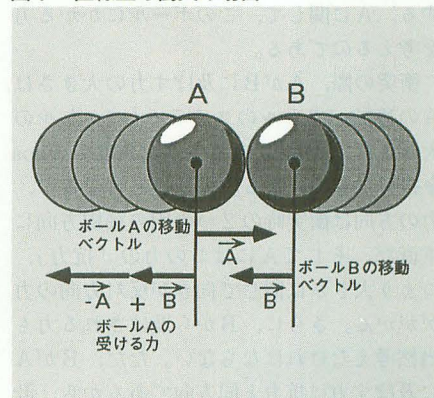
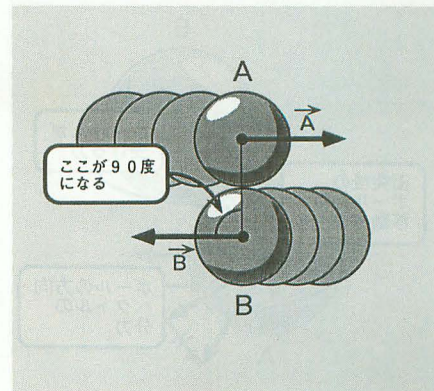



図2 ぎりぎりにすれ違う場合

直である」

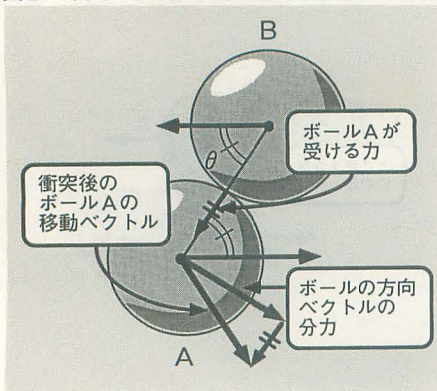
いまは便宜上ボールを二次元として扱っているので，これを「2つの円の接線に垂直」といい換えてもいい。

少し話を戻そう。どうして図1のようなときは相手のボールに対して100%の力がかかり，図2の場合には力はまったくかからないのだろうか。セオリーどおりにいけば，図2の場合にも進行方向と垂直な方向に力がかかっているはずなのではないか。これは「接線の垂直方向」という力の向きのほかに力の大きさを決定する要因が隠されている，ということなのだろう。

そこで注目するのが，図3の角θである。図1のように，2つのボールが正面衝突するときはθは0度だ。また，お互いをかすめてすれ違うときはθは90度となる。0度のときは100%で，90度のときは力がかからない，といって思い出すのは，三角関数cosである。どうやら衝突する2つのボールの間には，ボールの進行方向のベクトルに図2の$\cos\theta$を掛け合わせた大きさをもち，接線に垂直な方向の力が加わる，としてよさそうだ。あるいは，ボールの進行方向を表すベクトルを角度θで成分分解する，といったほうが表現としてはスマートかも知れない。

ここまでわかったところで，図1を見たときの複数のボールの反射の公式を一般化する。Aに関して，このボールにかかる力を考えるのである。

衝突の際，AがBに及ぼす力の大きさは，Aの移動ベクトルのスカラー(ベクトルの大きさ)に，図3にあるような角度θのcosを掛けたものになる。繰り返しになるが，力の方向は衝突時の2つの円の接線方向に垂直だ。そしてAにはこの力の「抗力」，つまり大きさは同じで向きが反対方向の力がかかる。さらに，Bから及ぼされる力も当然考えなければならない。ただ，BがAに及ぼす力は抗力と同方向であるから，計算はいくらか楽になる。

図3　平面上で衝突する場合

以上のことをまとめると，結局ボールAには，

$$|\vec{A}| \times \cos\theta + |\vec{B}| \times \cos\phi$$

の大きさをもち，2つのボールの接線方向に垂直で，Aの中心を向いた力がかかる，ということになるだろうか(ちなみに，いま考えているのは2つのボールの方向が平行なときだから，$\theta=\phi$となる)。

実際は，移動ベクトルのX，Y成分それぞれについて計算を行うことになるだろう。2つのボールが衝突したときの接点がわかれば，ボールの中心から接点に向かうベクトルが「接線方向に垂直なベクトル」となる。あとは接線方向に垂直なベクトルと移動ベクトルの内積を求め，ボールに及ぼされる力のベクトルを算出する。

そして，計算されたベクトルを用いて，衝突後のボールの移動ベクトルを弾き出すわけだが，ここでひとつ注意が必要だ。図1のような2つのボールが一直線上に並びながら衝突する場合，それぞれのボールの受ける力はボールの移動ベクトルに直接足していた。ところが，角θとϕがさまざまな値を取る場合を想定することで，移動ベクトルの分力を考えたのであった。分けた力のうち，ひとつは衝突した相手のボールに与える力だ。つまり，ボールの受ける力を足し合わせる相手には，分力のうち残っているほうを取らなければならない。そうでなければエネルギー保存の法則が成り立たず，計算結果がおかしなものになってしまうからである。

今回のサンプルは，4つのボールが画面上を跳ね返りつつ飛び回る，というプログラムである。詳しくはあとで触れるが，当然上記の方法を使ってボール同士の反射や衝突をシミュレートしている。ちょうどボールの止まらないビリヤードみたいなものと思っていただければいい。

ところで，図1のような状況で，片方のボールが止まっている場合はどうなるか，について少し考えてみよう。「2つのボールの移動ベクトルが交換される」のだから，動いていたボールは止まり，止まっていたボールが動き出す。ところが，ビリヤードで止まっているボールに白玉を当てると，白玉のスピードはかなり奪われるが止まりはしない。こうしてみると，いままで述べてきたシミュレーションの図式は完璧というわけではなさそうだ。

ボールのような球体が動くのには，2つの場合がある。ひとつは転がる場合，そしてもうひとつは滑る場合だ。これまで述べてきたのは，後者の場合，球体が摩擦ゼロの滑らかな平面を滑っていくときのシミュレーションなのである。転がって動く場合には，衝突によって抵抗を受けてボールの転がりは弱まるが，完全には止まりはしない。だから，ビリヤードの白玉は動き続けるのだ。その証拠に，白玉を弾くときにキューを突きぬかず，白玉を押し出したらすぐ引き戻すようにするとボールの回転は抑えられ，ほかの玉に衝突したときにピタリと止まるようになる。また同様に，白玉を突く位置でも回転を制御することができる。パソコンなどのビリヤードシミュレータで白玉を突く位置を指定できるものがいくつかあるが，これは玉の回転を反射計算などに反映させ，シミュレーションをよりリアルなものにする，という意図があってこうするものだと思われる。

ちなみにピンボールでは，ボールが衝突するのはどちらも動いている状況がほとんどだろうから，ボールの回転まで考慮に入れる必要はない，と個人的には思っている。ただし，ビリヤードの場合は止まっている玉との衝突が頻繁に起こるので，さすがに玉の回転を考慮しないとヘンになるかもしれない。

2階建て構造とクリッピング

「2階建て構造」というより，「立体交差」といったほうがわかりやすいかもしれない。現実のピンボール台には，ボールが跳ね回るステージに狭い坂道があり，勢いよく坂を駆け上がらせるとステージより高い位置にあるレーンをボールが通り，フリッパー横のランプに戻ってくる。もっと凝ったものになると，ステージの上に別のステージがあったりして，なかにはフリッパーまでついているものもある。これらをひっくるめて「2階建て構造」という。

さて，この2階建て構造を内部的に表現する方法について考えてみよう。まず問題になるのが，ボールが上にあるか下にあるかで，反射の処理が変わってくることだろう。たとえば，下のステージには障害物がある場所でも，上にはボールをフリッパー近くまで戻すレーンがついてる場合がそれにあたる。

しかしよく考えてみると，これは台が2階建てなのだから，裏マップも2つ用意すればいいだけの話である。残った問題はボールの位置が上にあるか下にあるかを判別する方法だが，これもさほど難しくない。というのは，ボールはかならず坂道などを通っ

て階を移動するのだから，坂道の入口や登りきった場所にボールが来たら，上にあるか下にあるかの情報を入れ替えるような処理をしてやればいい。

2階建て構造を実現しようとするとき，もうひとつ考えなければならないのがボールのクリッピングだ。特にボールが下のステージにあるとき，上にあるものを避けてボールを表示しなければならない。

普通のX680x0でゲームを作ろうとするとき，BGのプライオリティを高くして，前面に来るような物体はBGに描いておくとか，ハードウェア的な機能でクリッピングが実現できてしまう。だがピンボールの場合，クリッピングされる領域がかなり広範囲に及ぶので注意が必要だ。このような場合には特殊プライオリティを使えば実現できそうだが，今回はパスしておく。

それに代わる手として，表示するボールのスプライトパターン自体をクリッピングする，という方法をとろうと思っている。クリッピング用のビットマップをひとつ用意して，そこからボールが表示される範囲を切り出してきて，スプライトパターンのビットが立っている部分には透明色を割り当てるようにするのだ。ピンボールの場合，動いているものは通常ボールだけなので，速度的なコストもあまり心配する必要はないだろう。

ところで，クリッピングの必要性はなにも2階建て構造を導入しなくても出てくる。最も身近なところでは，当たるとボールが弾かれる円形のバンパーにボールが近づいたときに，クリッピングを行わなければならない。円形のバンパーはたいていキノコのようにカサをもっていて，このカサの下にボールが入ると一部が見えなくなるのだ。そのほかには，図4のような場合も考えられるだろう。

図4　クリッピングが必要な場合

とにかく立体的な台を作りたいと思えば，ボールをクリッピングする場面は増える。クリッピングに必要な仕様さえ用意してしまえば，このあたりはデザイン的な作業に委ねられるのだろうが。

サンプルについて

ボール対ボールの衝突(そして反射)をシミュレートするサンプルを作ってみた。リスト1，2はCとアセンブラソースの2つである。それぞれコンパイル，アセンブルしてリンクしたのち実行すると，画面上をボールが跳ね回る。終わりたいときはなにかキーを押せばいい。

ボール同士の反射の計算を行う前に，衝突判定を行わなければならないわけだが，サンプルでは以下のような方法をとっている。

まず，画面上のすべてのボールに対して，お互いぶつかる可能性のあるような矩形範囲にあるかどうかを調べる。もしそのようなボールの対が見つかったら，今度はもっと厳密な衝突の判定を行う。あらかじめ配列に代入してあるボールの周囲の座標と衝突判定の対象になっているボールの中心座標を突き合わせ，ボール同士が触れているか否かを決定する。こうしてボールが接している座標がわかれば，この座標を2つのボールの接線に垂直なベクトルとして使い，衝突後の移動ベクトルを計算する。

この方法の欠点は，ボールの周囲に関してだけしか調べないので，なにかの拍子にボール同士がめり込んでしまったときには衝突判定がなされず「すり抜け」が起こる，ということ。例によってボールが(内部的には)1ドット動くたびに衝突判定を行っているので，この「すり抜け」は滅多に起こらない。ただ，画面の端に反射した直後であるとか3つ以上のボールが団子状態になっているときには，たまにボール同士がすり抜けてしまう。

さて，今回でピンボールシミュレータを作るためのノウハウに関してはだいたい触れた。あとは，細々したアルゴリズムやらプログラミング上の問題が残っているが，この種のことがらは実際に作業を進めないと見えてこない，という側面をもっている。要するに「あとは作るだけ」なのである。身を入れて頑張らなくちゃ。　　(つづく)

リスト1

```
 1: #include "stdio.h"
 2: #include "iocslib.h"
 3: #include "basic.h"
 4: #include "basic0.h"
 5: #include "graph.h"
 6:
 7: #define bnum     4          /*動かすボールの数*/
 8:
 9: typedef struct  b_parm {
10:         int     x,y;
11:         int     dx,dy,cx,cy;
12:         int     ccx,ccy,drx,dry,pl;
13:         } b_parm,*b_parmPtr;
14: typedef struct  r_parm {
15:         int     x,y,dx,dy,cx,cy,f,drx,dry;
16:         int     dxx,dyy,ddx,ddy,bx,by,ex,ey;
17:         } r_parm,*r_parmPtr;
18:
19: extern void cut_bitmap(UWORD *base_addr,long l_bytes,
20:                 long cx,long cy,long xl,long yl,
21:                 UWORD *buff);
22: extern void clip_sprite(int sp_num,
23:                 UWORD *sp_pat,UWORD *cl_buff);
24: void    main_loop(void);
25: int     move_ball(b_parmPtr);
26: int     bump(r_parmPtr,b_parmPtr);
27: int     bsub(b_parmPtr,b_parmPtr);
28: void    bcalc( b_parmPtr,b_parmPtr,int,int );
29: void    calc_parm(b_parmPtr,r_parmPtr);
30: void    init(void);
31:
32: b_parm  bp[bnum];
33: UWORD   ballpat[64] = {
34: 0x0000,0x0012,0x0000,0x1246,0x0013,0x7887,0x0036,0x8877,
35: 0x0148,0x7776,0x0277,0x7666,0x1387,0x6665,0x2456,0x6578,
36: 0x2356,0x5686,0x1445,0x5554,0x0344,0x5445,0x0135,0x4354,
37: 0x0025,0x5655,0x0013,0x4666,0x0000,0x2487,0x0000,0x0012,
38: 0x2100,0x0000,0x2531,0x0000,0x7366,0x3100,0x6546,0x8600,
39: 0x5646,0x5710,0x5656,0x6520,0x8555,0x6471,0x6535,0x5672,
40: 0x5544,0x5682,0x5454,0x5771,0x4345,0x5730,0x3443,0x5510,
41: 0x5677,0x8300,0x7788,0x3100,0x8861,0x0000,0x4100,0x0000};
42: UWORD   pal[16] = {
43: 0x0000,0x18C6,0x318C,0x4A52,0x6318,0x8420,0xA528,0xCE72,
44: 0xFFFE,0x4000,0xCE72,0x4010,0x0001,0x2D8C,0x5ECB,0x9FD5};
45: int     bmp[58][2] = {
46: 42,39, 42,38, 42,37, 41,36, 41,35, 40,35, 40,34, 40,33,
47: 39,33, 39,32, 39,31, 38,31, 38,30, 38,29, 37,29, 36,29,
48: 36,28, 35,28, 34,28, 34,27, 33,27, 32,27, 32,26, 31,26,
49: 30,26, 30,26, 30,25, 29,35, 28,25, 27,25, 26,25, 25,25,
50: 25,26, 24,26, 23,26, 23,27, 22,27, 21,27, 21,28, 20,28,
51: 19,28, 19,29, 18,29, 17,29, 17,30, 17,31, 16,31, 16,32,
52: 16,33, 15,33, 15,34, 15,35, 14,35, 14,36, 14,37, 13,37,
53: 13,38, 13,39 };
54: int     co[58];
```

```
 55: UWORD   buff[128];
 56:
 57: void    main()
 58:
 59: {
 60:         OS_CUROF();
 61:         init();
 62:         clip_sprite(0,ballpat,buff);
 63:         while( !B_KEYSNS() )
 64:         {
 65:                 main_loop();
 66:         }
 67:         OS_CURON();
 68:         screen( 2,0,1,1 );
 69: }
 70:
 71: void    main_loop(void)
 72: {
 73: int     i,j,k,l;
 74:         for( i = 0; i != bnum; i++ )
 75:         {
 76:                 move_ball(&bp[i]);
 77:                 if( i == 0 )
 78:                         SP_REGST( i,
 79:                         bp[i].x+8,bp[i].y+8,1<<8,3 );
 80:                 else
 81:                         SP_REGST( 0x80000000|i,
 82:                         bp[i].x+8,bp[i].y+8,1<<8,3 );
 83:         }
 84: }
 85:
 86: int     move_ball(b_parmPtr bp)
 87: {
 88: int     i,r = 0,r2;
 89: r_parm  rp;
 90:         if( (*bp).pl )
 91:                 (*bp).pl--;
 92:         (*bp).cx += (*bp).dx;
 93:         (*bp).cy += (*bp).dy;
 94:         if( (*bp).cx > 0 )
 95:                 rp.dx = (*bp).cx >> 16;
 96:         else
 97:                 rp.dx = (-(*bp).cx) >> 16;
 98:         if( (*bp).cy > 0 )
 99:                 rp.dy = (*bp).cy >> 16;
100:         else
101:                 rp.dy = (-(*bp).cy) >> 16;
102:         (*bp).cx %= 65536;
103:         (*bp).cy %= 65536;
104:         if( rp.dx == 0 && rp.dy == 0 )
105:         {
106:                 return( 0 );
107:         }
108:         calc_parm( bp,&rp );
109:         rp.cx = rp.f / 2;
110:         rp.cy = rp.f / 2;
111:         rp.cx = (*bp).ccx;
112:         rp.cy = (*bp).ccy;
113:         while( rp.ex > 0 || rp.ey > 0 )
114:         {
115:                 rp.cx += rp.dx;
116:                 if( rp.cx >= rp.f )
117:                 {
118:                         rp.x += rp.drx;
119:                         rp.ex--;
120:                         rp.cx -= rp.f;
121:                         if( bump(&rp,bp) )
122:                         {
123:                                 break;
124:                         }
125:                 }
126:                 rp.cy += rp.dy;
127:                 if( rp.cy >= rp.f )
128:                 {
129:                         rp.y += rp.dry;
130:                         rp.ey--;
131:                         rp.cy -= rp.f;
132:                         if( bump(&rp,bp) )
133:                         {
134:                                 break;
135:                         }
136:                 }
137:         }
138:         (*bp).x = rp.x;
139:         (*bp).y = rp.y;
140:         (*bp).ccx = rp.cx;
141:         (*bp).ccy = rp.cy;
142: }
143:
144: int     bump(r_parmPtr r,b_parmPtr b)
145: {
146: int     i,j,re = 0;
147:         (*b).x = (*r).x;
148:         (*b).y = (*r).y;
149:         for( i = 0; i != bnum; i++ )
150:         {
151:                 if( !bp[i].pl &&
152:                     &bp[i] != b &&
153:                     bsub(b,&bp[i]) )
154:                 {
155:                         re = 1;
156:                         (*r).x = (*b).x;
157:                         (*r).y = (*b).y;
158:                 }
159:         }
160:         return( re );
161: }
162:
163: int     bsub(b_parmPtr b1,b_parmPtr b2)
164: {
165: int     i = 58,j,k,x,y,xx,yy;
166:         x = (*b1).x;
167:         y = (*b1).y;
168:         if( x <= 8 )
169:         {
170:                 (*b1).dx = -(*b1).dx;
171:                 (*b1).x = 9;
172:                 (*b1).pl = 1;
173:                 return( 1 );
174:         }
175:         if( x >= 248 )
176:         {
177:                 (*b1).dx = -(*b1).dx;
178:                 (*b1).x = 247;
179:                 (*b1).pl = 1;
180:                 return( 1 );
181:         }
182:         if( y <= 8 )
183:         {
184:                 (*b1).dy = -(*b1).dy;
185:                 (*b1).y = 9;
186:                 (*b1).pl = 1;
187:                 return( 1 );
188:         }
189:         if( y >= 248 )
190:         {
191:                 (*b1).dy = -(*b1).dy;
192:                 (*b1).y = 247;
193:                 (*b1).pl = 1;
194:                 return( 1 );
195:         }
196:         if( x-(*b2).x <= 15 && x-(*b2).x >= -15 &&
197:             y-(*b2).y <= 15 && y-(*b2).y >= -15 )
198:         {
199:                 if( y > (*b2).y )
200:                 {
201:                         for( i = 0; i != 58; i++ )
202:                         {
203:                           if( x+bmp[i][0] == (*b2).x &&
204:                               y+bmp[i][1] == (*b2).y )
205:                           {
206:                                 xx = bmp[i][0];
207:                                 yy = bmp[i][1];
208:                                 bcalc( b1,b2,xx,yy );
209:                                 break;
210:                           }
211:                         }
212:                 }
213:                 else
214:                 {
215:                         for( i = 0; i != 58; i++ )
216:                         {
217:                           if( x+bmp[i][0] == (*b2).x &&
218:                               y-bmp[i][1] == (*b2).y )
219:                           {
220:                                 xx = bmp[i][0];
221:                                 yy = -bmp[i][1];
222:                                 bcalc( b1,b2,xx,yy );
223:                                 break;
224:                           }
225:                         }
226:                 }
227:         }
228:         if( i != 58 )
229:         {
230:                 return( 1 );
231:         }
232:         else
233:         {
234:                 return( 0 );
235:         }
236: }
237:
238: void    bcalc( b_parmPtr b1,b_parmPtr b2,
239:                 int x1,int y1 )
240: {
241: int     x2 = 0,y2 = 0,x,y;
242: double  f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,si,co;
243:         x2 = -x1;
244:         y2 = -y1;
245:         f1 = (double)((*b1).dx);
246:         f2 = (double)((*b1).dy);
247:         f4 = (double)x1;
248:         f5 = (double)y1;
249:         f9 = sqrt(f1*f1 + f2*f2);
250:         f8 = sqrt(f4*f4 + f5*f5);
251:         f3 = f9*f8;
252:         f6 = (f1*f4+f2*f5)/f3;
253:         if( f6 < 0 )
254:                 f6 = 0;
255:         f1 = (double)((*b2).dx);
256:         f2 = (double)((*b2).dy);
257:         f4 = (double)x2;
258:         f5 = (double)y2;
259:         f10 = sqrt(f1*f1 + f2*f2);
260:         f3 = f10*sqrt(f4*f4 + f5*f5);
261:         f7 = (f1*f4+f2*f5)/f3;
262:         if( f7 < 0 )
263:                 f7 = 0;
264:         si = (double)y1/f8;
265:         co = (double)x1/f8;
266:         x = (int)(((-f9*f6*co) +
267:             (f10*f7*-co))*0.95);
268:         y = (int)(((-f9*f6*si) +
269:             (f10*f7*-si))*0.95);
270:         (*b1).dx += x;
271:         (*b1).dy += y;
272:         (*b2).dx -= x;
273:         (*b2).dy -= y;
```

▶Macintoshの使いにくいキーボードを使っていたら,X68000のあの素晴らしいキーボードがしっくりこなくなってしまった。　中沢　巧爾(18)京都府

```
274:         (*bl).pl = 2;
275: }
276:
277: void     calc_parm(b_parmPtr bp,r_parmPtr rp)
278: {
279:         if( (*bp).dx > 0 )
280:                 (*bp).drx = 1;
281:         else
282:                 (*bp).drx = -1;
283:         if( (*bp).dy > 0 )
284:                 (*bp).dry = 1;
285:         else
286:                 (*bp).dry = -1;
287:         (*rp).x = (*bp).x;
288:         (*rp).y = (*bp).y;
289:         (*rp).drx = (*bp).drx;
290:         (*rp).dry = (*bp).dry;
291:         if( (*rp).dx > (*rp).dy )
292:                 (*rp).f = (*rp).dx;
293:         else
294:                 (*rp).f = (*rp).dy;
295:         (*rp).ex = (*rp).dx;
296:         (*rp).ey = (*rp).dy;
297:         (*rp).dxx = (*rp).dx;
298:         (*rp).dyy = (*rp).dy;
299: }
300:
```

```
301: void     init(void)
302: {
303: int      i;
304: double   a;
305:         screen( 0,0,1,1 );
306:         SP_INIT();
307:         for( i = 0; i != 16; i++ )
308:                 SPALET( i,1,pal[i] );
309:         SP_ON();
310:         a = pi()/29;
311:         for( i = 0; i != 58; i++ )
312:         {
313:                 bmp[i][0] -= 27;
314:                 bmp[i][1] -= 39;
315:         }
316:         for( i = 0; i != bnum; i++ )
317:         {
318:                 bp[i].x = rand() & 0x1f0+8;
319:                 bp[i].y = rand() & 0x1f0+8;
320:                 bp[i].dx = (rand()<<3) & 0x3fffff;
321:                 bp[i].dy = (rand()<<3) & 0x3fffff;
322:                 bp[i].cx = 0;
323:                 bp[i].cy = 0;
324:                 bp[i].pl = 0;
325:         }
326: }
```

リスト2

```
1: *
2: * void cut_bitmap(UWORD *base_addr,long l_bytes,
3: *                 long cx,long cy,long xl,long yl,
4: *                 UWORD *buff)
5: *
6: * メモリ上のビットマップから指定領域を切り出し、
7: * バッファに書き出すルーチン
8: *
9: *    引数              引数の意味
10: *
11: * base_addr     ビットマップの先頭アドレス
12: * l_bytes       ビットマップの1ラインのバイト数
13: * cx,cy         切り出す領域の左上座標
14: * lx,ly         切り出す領域の縦横のバイト数
15: * buff          ビットマップを切り出す先のアドレス
16: *
17:
18: include iocscall.mac
19:
20: base_addr equ   8
21: l_bytes equ     12
22: cx      equ     16
23: cy      equ     20
24: lx      equ     24
25: ly      equ     28
26: buff    equ     32
27:
28: sp_top  equ     $eb8000
29: sp_num  equ     8
30: sp_pat  equ     12
31: cl_buff equ     16
32: next_sp equ     $80
33:
34: .xdef   _cut_bitmap
35:
36: .xdef   _clip_sprite
37:
38: _cut_bitmap
39:         link    a6,#0
40:         movem.l d0-d7/a0-a2,-(sp)
41:         clr.l   a1
42:         IOCS    _B_SUPER
43:         move.l  d0,-(sp)
44:         move.l  base_addr(a6),a0 * base_addrをa0に
45:         move.l  l_bytes(a6),d0   * l_bytesをd0に
46:         move.l  buff(a6),a1      * buffをa1に
47:         clr.l   d1
48:         move.l  cy(a6),d1        * cyをd1に
49:         mulu    d0,d1
50:         adda.l  d1,a0
51:         clr.l   d1
52:         move.l  cx(a6),d1
53:         move.l  d1,d3
54:         andi.l  #7,d3            * d3に (cx mod 7)
55:         move.b  #255,d4
56:         asl.b   d3,d4
57:         eori.b  #255,d4          * d4にマスクを設定
58:         lsr.l   #3,d1
59:         adda.l  d1,a0            * アドレス計算終了
60:         clr.l   d1
61:         move.l  lx(a6),d1
62:         clr.l   d2
63:         move.l  ly(a6),d2        * lx,lyをそれぞれd1,d2に
64:         subq.l  #1,d2
65:         move.l  d1,d5
66:         lsr.l   #4,d1
67:         and     #15,d5
68:         beq     leap_word_array
69:         addq.l  #1,d1
70: leap_word_array:
71:         lsl.l   #1,d1
72:         subq.l  #1,d1
73:         move.l  a0,a2
74: top_of_loop_y:
75:         move.l  d1,d5
76: top_of_loop_x:
77:         move.b  (a0)+,d6
78:         lsl.b   d3,d6
79:         or.b    d4,d6
80:         eor.b   d4,d6
81:         move.b  (a0),d7
82:         rol.b   d3,d7
83:         and.b   d4,d7
84:         or.b    d6,d7
85:         move.b  d7,(a1)+
86:         dbra    d5,top_of_loop_x
87:         adda.l  d0,a2
88:         move.l  a2,a0
89:         dbra    d2,top_of_loop_y
90:         move.l  (sp)+,a1
91:         IOCS    _B_SUPER
92:         movem.l (sp)+,d0-d7/a0-a1
93:         unlk    a6
94:         rts
95:
96: *
97: * void clip_sprite(long sp_num,
98: *                  UWORD *sp_pat,UWORD *cl_buff)
99: *
100: * スプライトパターンをクリッピングしつつ定義する
101: *
102: *    引数              引数の意味
103: *
104: * sp_num        スプライト番号
105: * sp_pat        スプライトパターン
106: * cl_buff       クリッピング用ビットマップのアドレス
107: *
108:
109: _clip_sprite
110:
111:         link    a6,#0
112:         movem.l d0-d7/a0-a3,-(sp)
113:         clr.l   a1
114:         IOCS    _B_SUPER
115:         move.l  d0,-(sp)
116:         move.l  sp_num(a6),d0    * d0にスプライト番号
117:         move.l  sp_pat(a6),a0    * a0にパターンのアドレス
118:         move.l  cl_buff(a6),a1   * a1にビットマップのアドレス
119:         mulu    #next_sp,d0
120:         movea.l #sp_top,a2
121:         adda.l  d0,a2
122:         move.l  #31,d4
123:         lea.l   mask(pc),a3
124: top_of_loop_sp:
125:         clr.w   d0
126:         clr.w   d1
127:         move.b  (a1)+,d0
128:         move.b  d0,d1
129:         andi.b  #$f0,d0
130:         lsr.b   #3,d0
131:         andi.b  #$0f,d1
132:         lsl.b   #1,d1
133:         move.l  #$ffff0000,d2
134:         move.w  (a3,d0.w),d2
135:         swap    d2
136:         move.w  (a3,d1.w),d1
137:         move.l  (a0)+,d3
138:         and.l   d2,d3
139:         and.w   d1,d3
140:         move.l  d3,(a2)+
141:         dbra    d4,top_of_loop_sp
142:         move.l  (sp)+,a1
143:         IOCS    _B_SUPER
144:         movem.l (sp)+,d0-d7/a0-a3
145:         unlk    a6
146:         rts
147:
148: mask
149:         dc.w    $ffff,$fff0,$ff0f,$ff00
150:         dc.w    $f0ff,$f0f0,$f00f,$f000
151:         dc.w    $0fff,$0ff0,$0f0f,$0f00
152:         dc.w    $00ff,$00f0,$000f,$0000
```

▶秋葉原へ行ったとき，暑さをしのぐためにかき氷を買った。ひょいと横を見ると「スパII」の対戦をやっているガキ……いやお子様がいた。それがすごい熱中度で思わず見入ってしまい……手元にはいちご水が。あーおいしい。　吉川　和男(23)東京都

愛読者プレゼント

1 Mu-1 GS

2名

X68000用

5″2HD版　28,000円(税別)

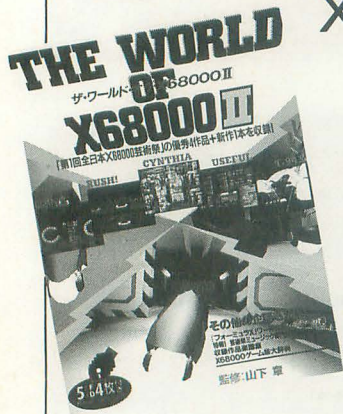
8月号でご紹介した音楽ツール。GS音源に対応したほか，さまざまなバージョンアップで，より使いやすくなりました。リアルタイム入力もできるので，楽器演奏ができる人には特におススメです。

▲サンワード　☎044(855)4335

2 The World of X68000 II

4名

X68000用

5″2HD版
4,900円(税込)

好評につき第2弾が発売されたディスク付きBOOK。X68000ユーザーのパワーを示すこの1冊，すでに重版となっています。詳しい内容は7月号のゲームレビューを見てくださいね。

▲電波新聞社　☎03(3445)6111

3 PC-VAN ウェルカムキット

30名

パソコン通信を始めてみたいけど，どうすればいいの？　という人にPC-VANの入会手引書をプレゼント。付属のユーザーIDでオンラインサインアップができます。

▲日本電気PC-VAN事務局

4 スーパーリアル麻雀PⅣ 原画&設定資料集

5名

ゲームだけではもの足りない！　なんてゼータクなキミに。あの3姉妹にもう一度会いましょう。原画や色設定だけではなく，ボツセリフ集や絵コンテなども収録。描きおろしイラストもあるのだ。

▲ソフトバンク　☎03(5642)8101

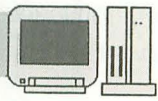
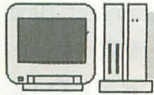
PRESENT

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1994年9月18日の到着分までとします。当選者の発表は1994年11月号で行います。また，雑誌公正競争規約の定めにより，当選された方はこの号のほかの懸賞には当選できない場合がありますので，ご了承ください。

7月号プレゼント当選者

1大魔界村　(栃木県)柏谷倫正　(東京都)古味貴徳　(大阪府)谷崎浩司　2F-Calc for x68k　(愛知県)竹内俊介　(大阪府)磯川輝千代　3Z-MUSICシステムver.2.0　(北海道)山田峰志　(東京都)佐井川泰治　(岐阜県)広瀬達也　(京都府)森田耕平　(大分県)藤林広幸　4CD「No Frame No Fame/X'Mass Song 68K」　(宮城県)遠藤弘樹　(新潟県)山中雅彦　(埼玉県)諸星城治　(神奈川県)林　慎明　(三重県)黒部浩孝　(京都府)北中詠司　(富山県)黒田博明　(石川県)小林　肇　(福井県)平木敬太郎　(岡山県)遠山幸男　ほか40名　(敬称略)

以上の方々が当選しました。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。

仮想ドライバの開発実験PART4.

仮想ドライブの運用実験とデータ収集

電机本舗 由井 清人 Yui Kiyoto

今回は，仮想ドライブシステムの問題点を洗い出してみます。思わぬところに落とし穴があったりして，一筋縄ではいかないようです。しかし，問題点を明確にすることで，信頼性の高い仮想ドライブシステムへの道が見えてきたようです。

今回は趣向を変えて，前回まで実験してきた仮想ドライブを実際に動かしてみて，さまざまな特性，信頼性をレポートしてみたいと思います。

レポート作成に利用した環境を表1に示します。

SX-WINDOWからの制御

今回のチェックは，筆者の利用しているSX-WINDOWver.2.0で行いました。一応，編集部にある最新のSX-WINDOWver.3.1でもチェックしたところ問題なく動いているようです(写真1の画面はSX-WINDOWver.3.1上での動作画面)。

下の写真1でPC-386側のドライブが，ドライブEとして表示されているのがわかるはずです。はじめドライブとしてSX-WINDOW上に出ていないので，心配した方もいるかもしれませんが，ドライブトレイの中には登録されています。ですから，ドライブトレイから，デスクトップへドライブアイコンを移せば，通常どおりに使用できるようになります。

SX-WINDOW上からの通常操作，ファイルコピー，リネーム，サブディレクトリの作成は，ほぼ問題なく動きます。ここで，ほぼ問題なくといったのは，妙に不安定な動作があるからです。これは，いろいろと測定器で調べたところ，SX-WINDOWだから不安定だというのではなく，PC-9801側でデータの取りこぼしが発生しているためでした。そして，データ再送，同調機構が思うように作動せず，ハングするという症状が出ます。

また，このようなわけで，ずいぶんと仮想側のFDのディレクトリ情報が壊れました。これは，再現性がほとんどなく，動作タイミングからくるものだと思います。ディレクトリ情報を更新しているときに，なんらかの理由でハングアップを起こしディスクが壊れるのでしょう。現在のシステムであれば，運用をFDに限定するのが好ましいと思われます。

オートイジェクトしてはいけない！

いろいろと操作してみて，いくつかの使用できない機能に気がつきました。

まず，必ずハングアップしてしまう，もしくは，やってはいけない操作というものがあります。

この筆頭が，ドライブアイコンについているディスク排出ボタンのクリックです。これを押すと，必ずハングアップします（写真2）。

はじめは首をひねったのですが，考えたらPC-9801のほうにはオートイジェクト機能がないので当然といえば当然なのかもしれません。通信監視測定器の伝える動作をみていると，SX-WINDOWとPC-9801は深い命令反復状況に入りハングアップします。永久ループに落ち込んでいるようです。おそらく，次のような事態が起きているのでしょう。

1) FDの排出を命令

写真1 ドライブアイコンの属性が「???」になっている

表1 テスト環境

X68000
1.機種 初代X68000
2.CPU＝MC68000，クロック＝10MHz，外部HDD＝20Mバイト1基，FDD=2HD2基，メモリ4Mバイト
3.Human68k ver.2.01

MS-DOSマシン
1.機種 PC-386noteW（エプソンのPC-9801互換機）
2.CPU＝V386，クロック＝16MHz，内蔵HDD＝40Mバイト1基，FDD=2HD2基，メモリ1Mバイト（DOSとして640Kバイト使用）
3.MS-DOS ver3.3B

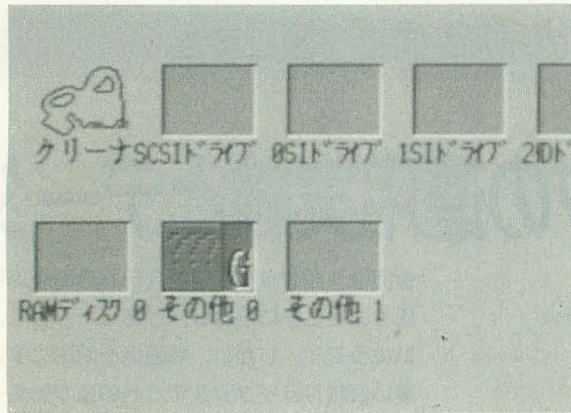



写真2　オートイジェクトするとハングアップするので注意

2)　FDが排出されているかPC-9801に確認
3)　排出されていないので（当たり前だ！），1)の処理へ戻り，繰り返す

FDの初期化ができない

SX-WINDOW上から（COMMAND.Xからも同じ），試しにFDの初期化を実行してみました。

結果は惨敗で，PC-9801上のディスクアクセスランプすらつかずに，「実行できませんでした」というメッセージが出てきます。

ここで写真1を思い出してください。仮想ドライブのアイコンは，ドライブのところに「？？？」がついています。これは，仮想ドライブがHuman68kの知らない形式のためにFORMAT.Xがただ単にエラーを出しているのだと思われます。面白いので，2HD（1.2Mバイト）と2DD（640Kバイト）のディスクをPC-9801にセットして，再びフォーマットを試みてみましたが，両方ともはねられてしまいました。

原因はいくつか考えられますが，たぶん，メディアバイトの指定の問題だと思います。覚えているでしょうか。ブロック型デバイスドライバは，BPBテーブルの中にメディアバイトと呼ばれる1バイトのデータをもっています。これは，デバイスの種類を表していて，FDであるとかHDDであるなどの識別コードとして機能しています。この識別コードがHuman68kと異なっているのです。

Human68kの利用しているメディアバイトの識別コードは，1バイト整数の負の領域に設定されています（80HからFFH，SRAMディスクで16進のF9Hです）。対して，MS-DOSは素直に正の数，1から始まる数のようです。実際に，7月号でPC-9801用の従機プログラム作成のおり，MS-DOSに対してDPB（Device Parameter Block）テーブルの取得を行いメディアバイトを調べたところ，1などの正の数字が格納されていたのを覚えています。このあたりの識別コードの違いが，動作の不安定さを招いているのではないかと考えられます。

FDを途中で交換してはいけない

仮想ドライブシステムでは，面白いことに2DDなどの本来X68000がサポートしていないメディアでも，従機側が対応していれば読み書きできてしまいます。ですが，ここで気をつけなければいけないことがいくつか発生します。たとえば，次のような場合です。

1)　途中で，FDが交換された
2)　同様にして，ただ交換されるのではなく，異なるメディア，2DDから2HDなどに交換された

このような場合を考えてみましょう。さらに，PC-9801は，本来ディスクの交換は手動で行うものですから，ディスク交換の識別はできません（最新型であればできるかもしれない。DOSのどこかでディスク交換属性を見た覚えがあります。はたして，機能しているかは不明ですが……）。

まず，現在の仮想ドライブシステムのメカニズムでは「R.EXE(R.X)」を起動しておき，主機側が電源立ち上げ時に1回だけ，従機側から仮想ドライブの情報を取得します。ですから，立ち上げ時のディスクの種類が主機側に登録されるわけです。要するに，異なる種類のディスクに交換することは，現行のシステムではやってはいけないことになります。

また，途中でディスクが交換された場合ですが，根本的に識別することができないため不都合が起きます。ですが，1回「CTRL+C」をコンソール上で実行すれば，Human68kはディスクのディレクトリ情報をすべて破棄して，取得し直しますから，限定つきではありますがディスク交換は行えます。

なお，今回試してみたメディアは表2のようなものです。

仮想HDDへアクセス

今回テストで利用したPC-9801互換機（エプソンPC-386noteW）は，内蔵の40MバイトのHDDを2つのパーテションに切って利用してみました。

このテストはさすがに恐ろしいのですが，PC-386のドライブをひとつ破壊する覚悟で実施してみました。現在，この機械は表3のように設定されています。

具体的には，全ドライブのうちの，BとCドライブを仮想化してみまし

表2　テストメディア

タイプ	容量
2HD	1.2Mバイト
2DD	640Kバイト
2DD	720Kバイト（/9フォーマット）

表3　従機側（PC-386noteW）のドライブ設定

ドライブ	種類
A:	HDD 20Mバイト
B:	HDD 20Mバイト
C:	FDD

た。そして，従機側のプログラムは次のとおりに指定しました。

R-DB[Cr]

ハードディスクはSX-WINDOWの上できれいに認識されます。FDのときに「？？？」ドライブとなったのとは対照的に，ハードティスクとしてドライブトレイに表示されます。

ファイルのコピー（古いVS.Xをサンプルにしてみました）とリネーム，およびディレクトリの作成を試してみたところ，FD同様にうまく動きました。

当初の予想では，はじめから全然動作しないと考えていました。これは，予想外とはいえ奇妙です。

今回テストに使用していたPC-386は内蔵のHDDを購入当初は1ドライブ40Mバイトに定義していました。古いMS-DOSのプログラムではHDDをうまく認識してくれなかったのが，20Mバイト以下のHDDでは動作するので，20Mバイトに分割した記憶があります。このあたりの事情が幸いしているように思われます。

COMMAND.Xからの制御

SX-WINDOWの上では機能がほぼ正常に動いているので，今度はCOMMAND.X上で，ひととおりの実験をしてみました。次に，動作確認した命令を述べます。

1) COPY
2) REN
3) CHKDSK
4) ディレクトリ操作

COPYコマンドは「VS.X」を仮想ドライブに転送してから，再び，本体のディスクにコピーしてみました。そして，おおもとのファイルとファイルコンペアを行い，等しいことを確認しています。

REN,CHKDSKはいうまでもないでしょう。通常のディスクと同じように動作しました。特にCHKDSKを2DDのディスクにかけて，正しく出たときは奇妙に感動します。なにしろ，Human68kにないディスクの種類が出てくるわけです。

ディレクトリ操作は，サブディレクトリの作成削除，カレントディレクトリの移動を行いました。

X68000同士の接続について

今回はX68000同士の接続についての詳しいレポートは割愛します。なぜかX68000同士の場合，SX-WINDOW上で利用できませんでした。オートイジェクトあたりのX68000とPC-9801の仕様の異なるところで，障害が起きているようです。

COMMAND.X上ではほぼ問題なく動いているようです。前回作成したPC-9801用の従機プログラムは，実はわずかな変更でX68000でも動作するようにできています。これを使って動作実験をしたところ，SX-WINDOWではこのままでは動かないとの結論に達しました。もう少し追求してから報告しようと思います。

ただ，COMMAND.X上での動作だけを見ていると，PC-9801に接続した場合よりは，かなり良好であると感じました。通信測定器で見ている範囲では，通信のデータの取りこぼしがほとんどなく，きれいに制御が流れているのが見えます。

次回予定

今回の実験で，今後の課題がある程度見えてきたと思います。まず，信頼性の向上。これは，通信部分で頻繁にデータの取りこぼしが発生しているのが原因だと思います。PC-9800のBIOSコールを利用している限り回避できないのではないかと推測しています。このあたりを書き直してあげれば信頼性を十分確保できそうな気がします。

次に，異なる仮想ドライブで，FDを抜き差し交換したときの回避処置の考慮が必要と思います。これについては，仕様として固定してしまい，活用方法できちんと禁止してしまうのも合理的かもしれません。

また，前回考えたとおり，高速化を含めて実験をしていきたいと思います。

そのほか，やってはいけないこと

本文で，やってはいけないことをある程度述べてきました。

ここでは，さらに基本的なことを述べます。当然といえば当然なのですが，MS-DOSとHuman68kではファイル名の利用規則が異なります。

まず，Human68kはファイル名の大文字小文字を区別します。これは当然，小文字の使用を許します。対して，MS-DOSは大文字小文字を区別せず，すべて大文字として扱います。

ですから，仮想ドライブシステムを利用して，小文字混じりのファイルをPC-9801へ送ることもできます。Human68kからは，仮想ドライブとして，小文字を名前にもつファイルを制御できます。

しかし，このようにして，PC-9801上に作られたファイルは，MS-DOSからは利用できません。少なくともファイル名が小文字で始まるファイルをMS-DOSは認識しません。このあたりをよく理解しておく必要があります。これは，仮想ドライブシステムの問題ではなく，異なるオペレーティングシステムの相違にかかわる問題点です。

また，Human68kはMS-DOSで予約しているファイル管理領域を利用して，ファイル名を最高21文字に拡張しています。ファイル名18文字の拡張子が3文字までです。MS-DOSでは，ファイル名が8文字に拡張子が3文字です。このあたりは，一般常識として理解しておく必要があります。

基本的にコンピュータの世界は「ない袖は振れない」のです。Human68kが名前を10文字拡張しているのは，MS-DOSが予約領域として確保しているスペースを使用しているからです。ですから，もし，MS-DOSがここを参照することがあれば，明らかに障害となって現れることになります。

新しい学部と100台のMacintosh

情報文化学とは？

この4月から名古屋大学の情報文化学部というところに転任しました。「情報文化学部」という名前から皆さんはどのような学部を連想されるでしょうか？　文学部，工学部，経済学部などのようにパッとイメージが湧く人はたぶんいないでしょう。

最近は，「人間」とか「情報」「環境」「文化」「国際」というような，重要であることを否定できないことばを2つ以上組み合わせて，学部や学科の名前にするというのが一種の流行のようです。

確かにそのような名前はあと何百年たっても通用するような立派な名前であることには間違いないのでしょう。ですが，「おお，耳ざわりのいい学部だ。で，いったい何をやるの？」と，その内容がつかみづらいのが最大の欠点といえるでしょう。あまりにも抽象的なのです。

情報文化学部を英語にすると，「School of Informatics and Siences」ということになっているようです。情報学と科学に関する学部ということです。ここでの「科学」は人文科学も含む広い意味で使われているものと考えられます。

英語という言語はあいまいさを許さないきわめて論理的な言語です。そして，情報文化の「文化」ということばがそのまま英訳されていないというところは，ミソかもしれません。

作家の石川好氏のいう「文化とは集団や国の勢力が強いとそれに応じてそれに伴うさまざまなものに対して外部の人が称するもののことである」という見方がひとつあります。それが正しいかどうかはおいておくとしても，文化ということばが外部からの評価，あるいは結果としての評価という面を少なからずもっていることは事実ですので，あいまいさを生ずるのはしかたのないことかもしれません。

「情報文化学部とは何を学問するところなのか」「どういう卒業生を生み出そうとしているのか」という質問をよく受けます。これに対して，たとえ，新設の学部で，この4月に新1年生が初めて入ってきたという状況であるにせよ，自分の学部を正体不明のようにいうのは無責任といえます。

そこで，少なくとも学部としてどのような方向を目指しているのかということを，公式の資料から引用しておきましょう。

「情報を単に定量的な抽象概念としてばかりでなく，意味や価値，作用力を備えた本来の姿において把握することが必要……自然や社会のシステムに内在する情報や意味のあり方の解明を基礎に，多岐にわたる諸科学を情報を軸に再構成し，より広い情報に関するソフトサイエンスを確立する」

「情報処理の技術を身につけるとともに，高度情報社会の諸課題を発掘し解決しうる幅広い視野と洞察力を養い，価値ある情報を選択し，新しい情報を創造・発信できる創造力，すなわち真の情報リテラシーを備えた人材の育成を目指す」

新入生全員にMacintoshが！

実際のところ，情報文化学部をひとことで説明するのは非常に難しいことです。まず基本的な分類である理系/文系という区別ができません。あらゆる分野の先生方がごちゃまぜになっています。そもそもこの新学部の母体が教養部であることから，これは当然なことともいえますが。

この学部には，自然情報学科と社会システム情報学科という2つの学科があり，それぞれを理系/文系と分けて考えることもできます。しかし，そういう従来の理系/文系の区別をしないで，情報という統一した観点からの総合性を重視するということがこの学部のウリで，両学科が深く連係したカリキュラムになっています。

また，実際，多くの数学の先生方や，僕のような計算機科学ずっぽりの人間も社会システム情報学科に属しています。数学については，入試科目として両学科に課せられるようです。

ところでそろそろ本題にはいっていくわけですが，この学部には大きなセールスポイントがあります。それがパーソナルコンピュータを使った情報リテラシー教育です。要するに，学生がパソコン(高学年ではワークステーション)を自在に使いこなせるようになるために，いろいろな面で手厚い配慮がなされているのです。

いま，大学生の皆さんは夏休みの最中だと思いますが，情報文化学部のピカピカの学生たちの多くは自分の家に大学のMacintosh(PowerBook)を持ち帰って自由に使っているはずです。

と，ごく簡単に書きましたが，実は意外と大変なできごとなのです。

授業料の高い私立大学では，学生にパソコンを買わせたり，貸与したりということは，そう珍しくはないと思いますが，国立大学で大学のパソコンを学生個人に貸し出すというのは，いままではなかったことです。とにもかくにも，先生方や事務の方々が奮闘されたおかげで，なかなか恵まれた環境が実現したというわけです。

PowerBookが100台ちょっと並んだ部屋が新たに作られまして，そこに学生がカード式のキーを使って入り，自分専用のPowerBookを使うことができます。そして，夏休みなどにはそのマシンを自分の家に持って帰ることもできるのです。

さらに，PowerBookのほかに，Quadra 840AVが1台おいてあって，マルチメディアシステムということで，さまざまな入出力機器がつながっています。

入力としては，イメージスキャナ，フィルムスキャナ，ビデオカメラ，8mmビデオ，VHSビデオなどがあります。出力は，フルカラープリンタ，レーザープリンタ，プロジェクタ，大型モニタ，あと一般的な名称はよく知らないのですがスライドフィルムに落とす装置などです。そして，そのQuadraには，さまざまなソフトがインストールされています。

これだけあると，ついPhotoshop(イメージデータを加工する高度なソフト)などで遊んでしまいたくなりますよね。いずれにせよ，Macintoshの好きな人にとってはよだれの止まらない環境が，この学部には実現されているといってよいでしょう。

せっかくのマシン環境ですから，講義でもこれを有効に活用するためのサポートをします。「プログラミング序論」という講義では，パソコンの類にまったくさわったことないという初心者も対象に，簡単な操作から，ワープロやお絵描きソフトの使い方などを教え，最終的に，自分の写真をスキャナで取り込んだりして自由に自分を表現する文書を作成するまでを教えます。

また，「知的生産論」という講義では，イ

ンスピレーションという発想支援ソフトを例として用いながら，マシンを利用してどのように発想を行うかを教えます。また，「外書講読」では，チューリングマシン，あるいは，一階述語論理の概念を専用のソフトウェアを使用して教えます。

僕は「プログラミング序論」におけるMacintosh関係を担当しています。状況をひとことでいえば，四苦八苦しているといったところでしょうか。ハードやソフトの問題は，まあ基本的にはがんばればなんとかなるというような問題が多いのですが，いちばん苦労しているのが情報文化学部の特殊性です。すでに述べたように，何を目指しているのかわかりやすい歴史のある学部とは違うので，入ってきた学生もまた教官と同様にきわめて多様であるという事実があるのです。

たとえば，Macintoshの使い方をわかりやすく教えていても，わかっている学生は暇そうにしているし，初めて体験する学生は案外苦労しているのです。もちろん，工学部の学生にしてもそのようなばらつきは当然ありますが，それがあまりに際だっているのです。したがって，教える側にとっては，どのあたりを想定して講義を構成すればよいのかということで悩むのです。

Macintoshで何をする?

我々がそれなりに努力して作りつつあるこの環境なのですが，学生もパッと見には喜んでくれているようです。また，こちらも新しい環境をひとつずつ作っていく実感があります。これらは教官にとって十分なことといえます。

ただ，よく思うことは，将来的には，基本的な計算機リテラシー(漢字変換とか，写真などを文書に取り込む技術など)は，小学校や中学校で教えることになるだろうということです。また，ネットワークを通じた使い方(メイルのやりとりや，ファイルシステムの共有，ニュースの読み書きなど)は，高校ぐらいで教えることになるだろうということです。

それが実現するためには，従来の小・中学校や高校のカリキュラムの一部を圧縮したり，削除したりする必要があります。それが何かということについては，現状をあまり知らないので即答できません。ですから，この予想は，責任あるものではないかもしれません。

illustration：Haruhisa Yamada

しかし，いずれにせよ，時代が進むにつれて，学校で教えなくてもかなりの情報リテラシーに関する知識が若い人に伝わるようになっていくのは間違いないでしょう。そうなると，いまここでやっているようなことの大部分はわざわざ大学で教えなくてもすむだろうと判断しているのです。

そのようなときに，それ以前の基本的な情報リテラシーに引き続く教育として大学は何を教えるべきなのか？　ということが問われてきます。

それに対するひとつの答えが，この学部で用意されている「知的生産論」的な方向性であると考えられます。つまり，もう少し創造的な仕事(たとえば文章作成)に計算機を役立ててもらおうというわけです。

僕はこの講義の内容についてはあまり知らないのですが，インスピレーションというソフトを利用していると聞きました。そして，この数カ月，僕はそのインスピレーションを使っています。

このソフトは，いわゆるアウトラインプロセッサ(文章を作る前段階において，章立てなど全体構成をどうするかの構想を練るためのソフト)的な性格をもつだけでなく，浮かんだアイデアや概念を断片的でもいいから書いていき，それらの間を矢印で関連づけることにより，徐々に全体のアイデアをまとめていくというアイデアプロセッサ的性格ももっています。

そして，このソフトでは，その2つの使い方を自由に切り替えることができます。最初は発想を自由にアイデアプロセッサでお絵描きソフトのようにして平面上に作っていき，次に，文章化を意識しながら，アウトラインプロセッサも使っていくのでしょう。そして，プレゼンテーション用ならば，最終的にアイデアプロセッサのほうで図を作成することになりましょう。

僕はといえば，まだ，完全に慣れてはいないのですが，はっきりいって「挫折」気味です。アウトラインプロセッサのほうは昔，Moreというソフトを使い込んでいたこともあり，十分使えそうなのですが，せっかくのアイデアプロセッサのほうをほとんど使う気がしなくなってきたのです。

単に字を並べるワープロの次にくるのは，構造を編集できるアウトラインプロセッサ，そして次には発想の段階からサポートするアイデアプロセッサであるという確信は揺らいでないのですが，少なくとも自分で使いこなせないようでは話になりません。単にソフトに改良の余地が多いのか，根本的なところに問題があるのかはっきりしません。しかし，どうも馴染めないのです。

そういうわけで，僕自身，いまやっている教育をもっと早い時期にすべきだとは強く感じつつも，代わりに何をどのようにすべきかということに関してはまだはっきりとした答えをもっていない状態です。

計算機科学とか，工学ならば，まだしもそこで行う教育も方針をたてやすいのですが，なにせ，「多岐にわたる諸科学を情報を軸に再構成し，より広い情報に関するソフトサイエンスを確立する」ための計算機リテラシーなのですから。

石の言葉，言葉の夢

［第6回］

PDAは液晶ビューカムをめざす

Ogikubo Kei
荻窪 圭

ロンドンへ行ってきたのである。愉快なロンドン楽しいロンドンのロンドンである。時差が9時間もあるロンドンである。日の出の割に日の入りがやたら遅いと思ったら，サマータイムとかで1時間ずれていたというロンドンである。

目的は観光だ。観光。で，ふらふらと観光してきた。サッカーゲームがやたら氾濫していたとか，ゲーム雑誌の付録にワールドカップの勝敗表(もちろん,結果は読者が自分で埋めていくのだ)がついていたとか，とにかく，ウインブルドンとツール・ド・フランスとワールドカップで盛り上がっていたロンドンである。ついでに，クリケットもテレビで放映されていたな。見たけど，何が何やら全然わからん。クリケットって，いったい何だ？

ちなみに，ロンドンの電気街(AV機器ショップとパソコンショップが並ぶ通り)は大英博物館の近くにあるトッテナムコートロードってとこ。行くと結構楽しい。

あっちへ行って思ったのだが，凄いね，向こうの人は。人と話すとき，絶対に目をそらさない。こっちの目をじっと見て喋る。こちとら，見つめられて緊張するうえ，英語なので何いっているかわからなくてさらに緊張する。困ったものだ。

じっと目を見る。これ，結構重要なことだ。しかし，である。よくあるテレビ電話だとかテレビ会議のシステムって，相手の目をじっと見られないのだよね。あれは気持ち悪い。相手の顔が伏し目になる。当たり前で，こちとらモニタに映っている相手の目を見ているのだが，それを捉えるカメラはたいていモニタの上についているからだ。すると，相手にはちょっと上から撮った映像が送られる。

アメリカのニュース番組では，キャスターが視聴者に向けた視線をそらさなくて済むように，テレビカメラのレンズ方向にハーフミラーか何かで原稿が映る，って話を聞いたことがある。そうすれば，カメラ目線で原稿を読めるわけだ。

テレビ電話の最大の難点はここにある。カメラ目線であり，なおかつ相手の顔も見られるシステムをどう作るか。

視線が来ているかいないか

カメラ目線を保ちつつモニタに映った相手の顔を見るにはどうしたらいいか。要するに，視線上にカメラを置けばいい。マジックミラーだかハーフミラーだかを使って擬似的に視線を一致させてもいいが，それは大がかりで，家庭用の機器には向かない。

私が提案するのは，もっと単純。文字どおり，モニタと顔の間にカメラを置けばいいのである。邪魔じゃないか，と，思うのが人情であるが，そうでもない。CCDカメラなんてせいぜい直径3cmくらいの箱に収まるものだし，カメラはフレキシブルアームかなんかで本体に固定しておいて，必要なときだけ，モニタの前にひっぱってくればいいのである。

たとえば，モニタの前10cmくらいのところに3cmほどのカメラがあるとすると，人間の目って2つあるから，なんとかカメラの向こうのモニタを見渡すことができる。重要な映像ならキャプチャしてゆっくり見ればよい。カメラの位置を自在に動かせるようにすれば，相手の顔だけでなく，ほかのものを映して送るのも簡単だ。

電話するときは，カメラをモニタの前に持ってきて対話し，ほかに送りたい映像があれば，それをハードディスクから送信するか，カメラをほかの方向に向けてそれを映せばいい。

細かい話ではあるけど，ユーザーフレンドリーっていうか，使う人の感覚を重視するインタフェイスってのは，案外，こういう細かい工夫からなっているのではないかしら。

めざせ液晶ビューカム型PDA

CCDカメラといえば，液晶ビューカムである。

昔，どっかの雑誌に書いたのだが，液晶ビューカムは，究極の携帯コンピュータの姿なのである。TFTのカラー液晶パネルがあり，CCDカメラがあり，マイクがあり，スピーカーがある。映像・画像・音声の入出力ができる。バッテリ駆動ができる。あとは文字やポインティングなどのOS操作だ。それは，ペンで行えばいい。

記憶デバイスは，そうだな，8mmビデオテープってわけにはいかんから，TypeIIIのPCMCIAカード型1.8インチハードディスクなんかがいい。容量は，まあ，1Gバイト。それがムチャなら200Mバイトくらい。ちなみに，現在発表されているのは，170Mバイトくらいだったかな？これがリムーバブルメディアで，もちろん，内部にはフラッシュメモリだかの記憶デバイスが入っている。

大きさは，いまの液晶ビューカムよりちょっと液晶モニタを大きくして(いまのは4インチだけど，まあ，6インチはほしいところだ)，テープ駆動部分がなくなっただ

け，薄くする。
もちろん，通信機能は標準装備。FAXモデム機能は当たり前として，PHS対応でデジタル通信も可能。もちろん，そのまま電話としても使える。携帯電話ではなく，パーソナルハンディホンだ。
ゲーム用に十字キーパッドとボタンがついているのもいい(何考えてんだか)。
それで，バッテリ駆動8時間(CCDカメラをOFFにした場合)。CCDカメラは映像を撮るより，デジタルスチルカメラとして使うほうが多いだろうな。
CCDカメラというと，画質が気になるところだが，本体に直結すれば，いったんアナログのNTSC信号に落とす必要もないから，結構な画質が保てるのではないか。
重さは700gくらいに抑える。
とまあ，このくらいになるといい。
デスクトップ機とはPHSを使った無線LANで結ばれており，相互の情報はお互いにやりとりできるようになっている。
OSは，そうだな，Windowsはやだけど，使いやすければ，ザウルスみたいなのでもいい。ジェネラルマジックのMagicCapみたいなのでもいい。スクリーンのビデオカメラアイコンをタップ(ペンで画面を軽く叩くこと)すると，モニタウインドウが出て，録画状態になる。スチルカメラアイコンをタップすると，静止画キャプチャ状態になる。テレビアイコンをタップすると，録画したデータを再生できる。電話をタップすると，パーソナルハンディホンになる。
ノートをタップすると，ノートの画面になり，そこには文字でも絵でもボタンでも映像でも音声でも何でも自由に張り付けられる。
アプリケーションはノート上で動く。つまり，ノートがワークスペースになるわけ。Go社(いまはAT&Tに入っちゃったけど)のPenPointがそんな概念になっていた。
なかなか楽しいと思うのだが。実現されるのはいつになるのだろうか。

ハンズフリー電話になるパソコン登場

電話機になるパソコンはすでに出ている。
ん？　そういえば，10年近く前にもあったな，留守番電話になるパソコンが。MZ-2500とかいう。あれは本体についているカセットテープレコーダーに音声を録音する仕様になっていた。時代が時代だから仕方がない。
いまならサンプリングしてハードディスクにレコーディングできる。それをやったのが，オリベッティのQUADERNOとPHILOSである。オリベッティといえばイタリアのメーカー。さすがイタリア。考えることがヘンである。
QUADERNOとPHILOSはDSPを内蔵している。デジタル・シグナル・プロセッサのDSPだ。こいつが音声関係の処理を一手に引き受ける。FAXもモデムも音声電話もすべてDSPが行う。FAXやモデムの場合はDSPがエミュレーションを行う。音声の場合はリアルタイム圧縮を行うのかな。聞いたことがない方法だが，AD PCM以上に圧縮をしてくれる。
QUADERNOはA5サイズのサブノートパソコンだ。日本ではROMに入っている英語版Windowsに，Win/Vという英語版Windows上で日本語を使うためのソフトをかぶせて売っている。蓋を閉めるととてもパソコンには見えないデザインで，なんと，蓋には録音・再生などのボタンまでついている用意周到ぶり。そのまま，テープレコーダーとして使えまっせ，ってことだ。
PHILOSはA4サイズの普通のカラーノートパソコンだが，こちらもDSPを内蔵している。留守番電話にもできるし，FAX/モデムにもなる。非常に面白い設計だ。
QUADERNOなんて，あとCCDカメラがついてPHSに対応すれば，完璧だ。コードレステレビ電話対応パソコンになる。だってね，テレビ電話になっても(ほんとになるのだろうか，って不安はあるが)，コードレスホンの魅力には勝てないだろうから。A5サイズの中にどこまでツールを埋め込めるか。追求してみる価値はある。
そもそも，日本人の場合はテープレコーダーを持って音声メモを取るなんて習慣がないからピンとこないかもしれないが，欧米では結構面白い存在になるのかもしれない。
ロンドンへ行って思ったのは，みんなよく喋る，ってこと。とにかく，よく口を開く。パブへ行けば何話すことがあるのか，始終喋っているし，街を歩いていても「Excuse me」と「Thank you」と「Sorry」のオンパレードだ。人をかきわけて地下鉄から降りるときは「Excuse me」だし，ドアを開けて待っていてあげると「Thank you」だし。ウェイターもやたらとあいそよく話しかけてくる。
もともと，音声伝達の文化なのだなあと思う。日本だと結構目と目の挨拶だとか，軽い会釈で済ましてしまいそうなことばかりだ。いちいち声を出すので疲れるが，旅行者にとっては非常に気持ちがいい。
しかも，あいそがいい。あの胡散臭いわざとらしい笑顔はどうすれば真似できるのか知りたいほどである。
そんな文化だから，テープレコーダーに音声メモを取ったり(晴海や幕張でも，テープレコーダー片手にブースを回っている欧米人をよく見かけた)，マイクを内蔵したパソコンを出したり(MacintoshのPowerBookシリーズも本体にマイクを内蔵している)するのだろう。もっとも，パソコンに向かって音声メモを取っている人を見たことはないから，いったい誰がどのようにマイクを使っているかはわからないが。
日本だと，やはり，マイクよりCCDカメラが喜ばれそうだ。でもわからないな。人前で堂々と携帯電話を使う人も増えていることだし。携帯パソコンに向かって喋るっていう電話が違和感なく受け入れられる日がこないとも限らないのだ。通話をいちいちハードディスクに録音して言質を取られるような時代だけはきてほしくないものなのだけどね。

第94回

猫とコンピュータ

キーワードを増やそう

Takazawa Kyoko
高沢 恭子

旅人の地図や道標のように，探しものには手がかりが必要です。見つけたものは次の発見への手がかりになり……そうして世界は広がるのかもしれません。でも，ちょっとしたカンチガイにもご用心。

シシはライオンだった

「そんなことも知らないのぉ？」なんて，他人についてはよく思う。

テレビの対談番組のなかでとても高名な女性司会者が，「帰国子女っていうけど，男の子でもみんな『子女』になっちゃうのネ。あれ，おもしろいわね」といっているのを聞いた。「えーっ，『子女』とは『むすことむすめ』のことをいうのに，ほんとにこの人は知らないのだろうか」と，私はしばらく考えこんでしまった。

それでは，そんな識者きどりの私が人から笑われることはないのだろうか。いや，そうはいかない。うまいぐあいに誰にもわからなければまだいいが，わかってしまうこともやっぱりある。

幼い人や十代の人たちが「そんなことも知らないのか」の対象にならないのは，まだすごしてきた時間が短いからという当然の理由があるからで，勉強のためのじゅうぶんな時間を送ってきたはずのオトナは，もう言い逃れはできない。

トオルが中学生のころだったか，「お母さん，『シシフンジン』の『シシ』は『獅子』でいいんだよね」と聞いた。私は「シシフンジン」を頭のなかで漢字変換した。まちがえてはたいへんだ。「奮迅」はふるいたつことだけれど，「シシ」はライオンだったろうか。なんだかコッケイじゃないかなと急に思った。「ちがうんじゃない？手足をふんばってがんばるから，『四肢奮迅』でしょ」と答えた。びっくりしたトオルはすぐに国語辞典を調べて，苦笑いしながら正解の『獅子奮迅』を示した。母親のあまりの迷答がショックだったのか，さすがに大笑いはしなかったが，こちらは消え入りたいほどだった。

「そんなことも知らなかった」恥ずかしさの量は，いままですごしてきた時間の長さに正比例して自分に返ってくる。裏返しに洋服を着て外出し，それに気づくまでの時間が長ければ長いほど恥ずかしさが大きいのと似ている。できれば，このことを誰も知らなければいいと願う。「シシ奮迅」の一件については「誰にもいうな」とトオルに厳命したけれど，子供である自分のほうが百倍も恥ずかしくなるような話を口外するわけがない。

珍変換を生かす法

国立国会図書館に所蔵されている明治期刊行図書(全16万冊)が，数年前，丸善と国立国会図書館によってマイクロフィルム化された。この「国立国会図書館所蔵明治期刊行図書マイクロ版集成」のために，丸善の企画・監修で索引用CD-ROMがつくられたのだが，そのなかに入力ミスから書名，著者名，出版社名などのフリガナ部分の表記に誤りのあることがわかったという。

5月30日付の日本経済新聞によると，まちがいの例はつぎのようなものだった。「出埃及記」は「シュツエジプトキ」とすべきところを「デアイオヨキ」。「伯多大帝」は「ピョートルタイテイ」とするべきだが「ハクタダイテイ」。「弥児頓」は「ミルトン」を「ヤジトン」。

ほかにも，森鷗外の「西周伝」(ニシアマネデン)を「サイシュウデン」，「伽羅先代萩」(メイボクセンダイハギ)を「キャラセンダイハギ」などとしており，研究者の指摘で，60件以上のミスが見つかったそうだ。国際的にも注目されている出版物に関することなので，関係者は対応に苦慮しているとあった。

誤って入力された読みかたは，なんだか少し不自然な気もする。「出埃及記」という文字を人が書名らしく読もうとするとき，「デアイオヨキ」なんて読むだろうか。あるいは自動的に読み取るソフトがあって機械的なカナ変換をしたが，そのあとの人為的なチェックがふじゅうぶんだったのだろうか。16万冊もの数の書籍をリスト化するためにコンピュータが利用されないなんて考えられないから，なにかの機械的な処理が，ちょっとユーモラスなミスを生んでしまったにちがいない。とはいっても，古典や文献のタイトルのフリガナのまちがいは，文化遺産の尊厳にもかかわることだから，研究者のかたたちの緊張は当然だ。

ところで明治時代に刊行された書籍というと，文字も表記法も百年前に使われていたものであり，日本語ではあっても19世紀の国語である。専門家やくわしい知識をもった人には常識である書名でも，ふつうに現代の文字や言葉のなかですごしている人にとって，「出埃及記」はほとんど外国語に近い。ことわざや熟語は長く受けつがれているものだから，「獅子奮迅」を正しく読み書きできないと笑われるが，「弥児頓」を読めなくてもたぶん笑われない。

一般的な知識をもったある人が「出埃及記」について調べる必要ができたとする。すでに耳から聞いてこの書名の読みかたを知っていればいいが，そうでないとしたら，まずなんとかしてこの表題を読まなければならない。そういうとき，自分の知る範囲，想像できる範囲でいろいろな読みかたをしてみるだろう。そのなかにはきっと，「デアイオヨキ」や「シュツジンキュウキ」なんていうのもある。

読みかたのわからない文字を読むときは知ってることを手がかりにする。これは高

田任康さんの「漢ぺき君」の流儀である。なんとか暫定的に読みかたをきめて，とりあえずそれと同じ文字のある場所をさがしだすのだ。

「出埃及記」も「伯多大帝」も，とてもスラスラ読める書名とは思われないのだから，「デアイオヨキ」や「ハクタダイテイ」も検索語の1つとして掲載しておいて，それぞれ正しい読みかたを併記しておけばいいのではないか。索引はさがしだすためのリストなのだから，誤読でも見つけられるようなサービスがあってもいい。最終的に正確な資料やデータが得られることが索引の役割であるし，それには手段ができるだけたくさんあるべきだ。高田さんの「漢ぺき君」には，1つの文字に対して幾通りもの検索語が用意されている。

新しいキャンパス

いくつかの辞典で「子女」の語義を調べてみたら，「むすことむすめ」「こども」のほかに「おんなの子」の意味もあり，辞書によっては「おんなの子」が第一義になっているものもあった。「士女」が「紳士，淑女」をさすように，「子女」は「子息と息女」の意味だけだろうと思っていた私もまちがっていた。なにか疑問をもって調べるたびに，自分の知っていることがいかに少ないかよくわかる。

トオルの学ぶ大学には，その帰国子女と呼ばれる人たちの在籍がとても多いらしいのだが，トオルの学科も50名ほどの定員のうち3分の1くらいが，帰国子女なのだそうだ。

先日トオルが，「お母さん，東京と三重を往復してたいへんだね，2つの文化圏で混乱しない？」と本気とも冗談ともつかないようなことを聞いた。

「クラスの帰国子女の子たちはネ，自分はいったい，いままでいた国の人間，アメリカ人，あるいはイギリス人なのか，日本人なのかって，つまりアイデンティティーといったものに，深刻な不安を感じてるらしいよ」

海外でたくさんの感化を受けながら成長期を送り，勉学と青春のまっただなかに国が変わる。それはどんな気持ちだろう。

トオル自身が大学という新しい場を得たことも，それなりに大きな変革のようだ。勉強の主体がはっきりと自分になり，内容も方法も自分の意思と責任できめることができる。いまそれが新鮮であり，うれしくてたまらないのだそうだ。

「1年生から取っていい専門科目もあるし楽しいよ。外国人の先生で，日本語を話せるのに，はじめから終りまで英語しか使ってくれない先生がいてネ，そういうときはシンドイけどね」

illustration：Kyoko Takazawa

いろいろな国の先生も学生もいるし，帰国子女の友人たちもいるとなれば，トオルとしては，いままでの都内の地方区からいきなり全国区，世界区のキャンパスになったわけだ。楽しさもさることながら，じつはたくさんのカルチャーショックを味わいつつ，ひそかにセルフコントロールをしているのだろう。

東西文化交流

アイデンティティーの確立の悩みとはほど遠い私だが，2つの場所で交互に生活しているための小さなトラブルはたくさんある。それはとても具体的だ。

部屋の照明のスイッチやコンセントのありか。FAX，コピー機のとりあつかい。洗濯機，掃除機，電子レンジの操作。お風呂のわかしかた。文具や書類，貴重品，印鑑，カギの保管場所。2つの場所にあるそれらのものは，すべてがよく似ていてみんな少しずつちがう。

けっこう大きなちがいもある。テレビのチャンネルの番号がぜんぶちがう。東京の1チャンネルは三重では2チャンネル。4は6，6は10，10は4，8だけは8なのが救いで，これはいまだに混乱のさなかだ。

問題はチャンネルのナンバーだけのちがいではない。番組の編成が関西中心であることだ。それはあたりまえだが，1つだけ不満なのはプロ野球の実況中継で，巨人戦がカードによっては予備放送にまわることだ。巨人－阪神戦は放送されるけれど，巨人－横浜戦と同時に阪神－中日戦があると，テレビ，ラジオとも阪神－中日戦だけになる。試合が西寄りか東寄りかで微妙に変わるので，これがいちばんコタエる。

おもしろいのは，双方で同じ新聞を購読しているために，同一の特集記事などを掲載時期や紙面の編集を変えてふたたび目にすることだ。すぎ去った時間がくりかえされているようなふしぎな気がする。また，2つの土地の生活感のちがいが最もはっきりとあらわれるのは読者の投稿欄で，とても興味深い。

こうしてみると，いちばん変動のないのは，やはりパソコンということになるのだろうか。両方の場所で，同じソフトを使ってしごとをしたり絵を描いたりできる。同じ機種が双方に待機してくれているのはありがたい。パソコン通信のおかげはやはり大きく，原稿や情報のやりとりができるので，どこで生活しているかという条件はあまり意識のうちにない。

そしてもう1つ，かならず新しい土地でパソコンのプロフェッショナルに出会うことだ。とうとう，数日ちゅうにわが家にもPC互換機とWINDOWSがやってくることになった。

Another CG World
no 37
©KYOKO EGUCHI

ふたば…

何しとん?…

ふーん

影絵かあ〜
そーいえば小さいとき
よく遊んだなぁ〜

わし…

はくちょう…
影が黒いのに
なんでハクチョウ
なの?と思った
りした…

うさぎ…

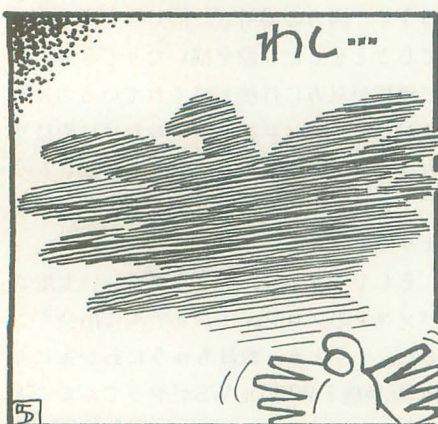
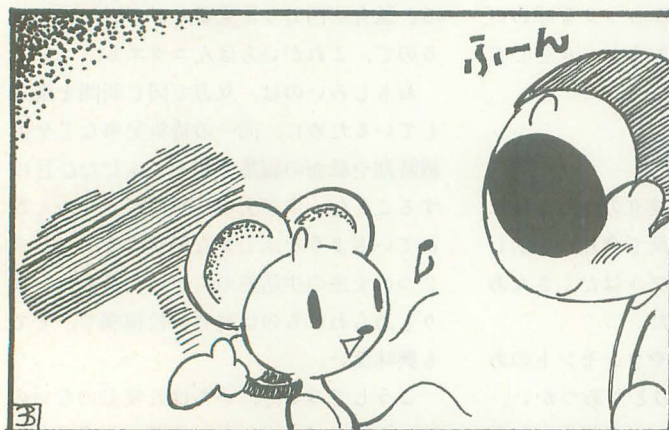
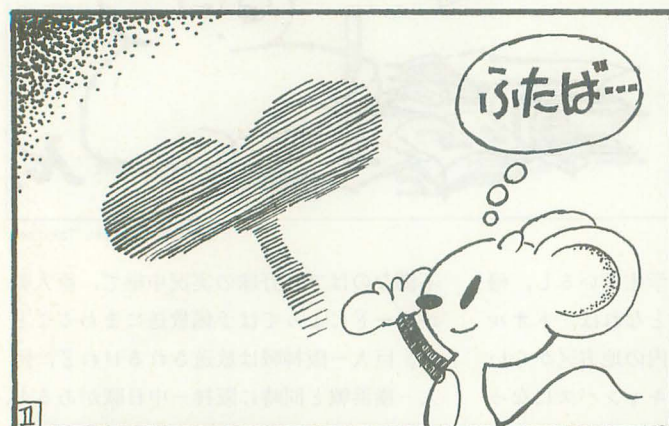
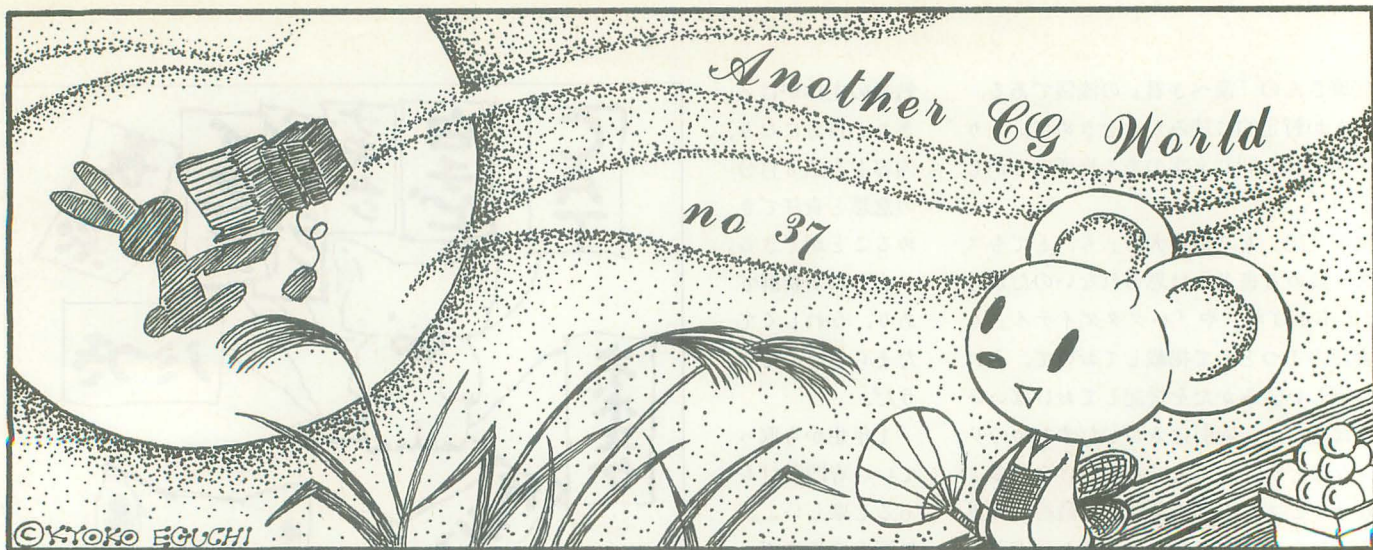
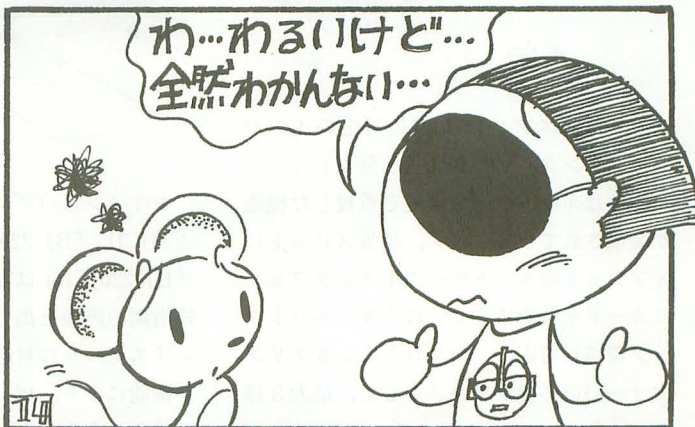
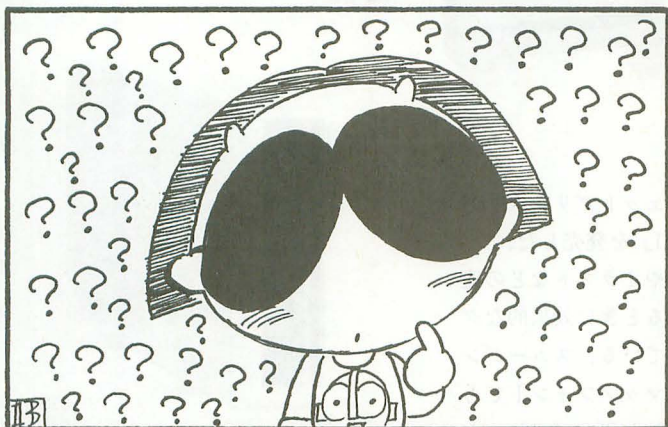
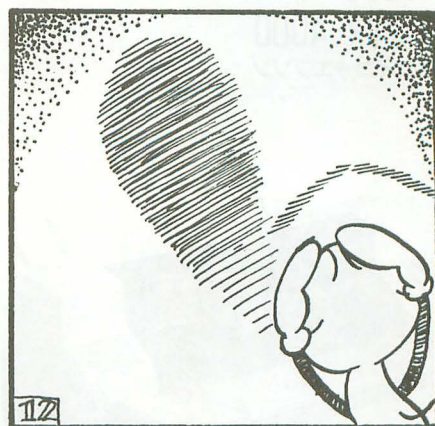
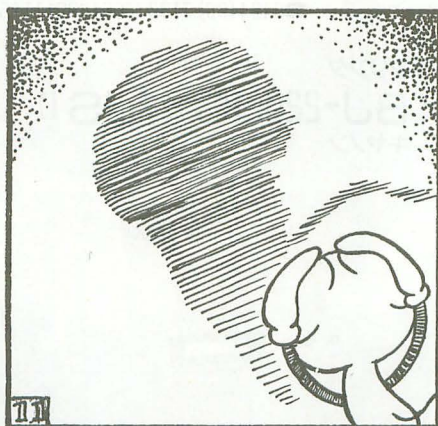
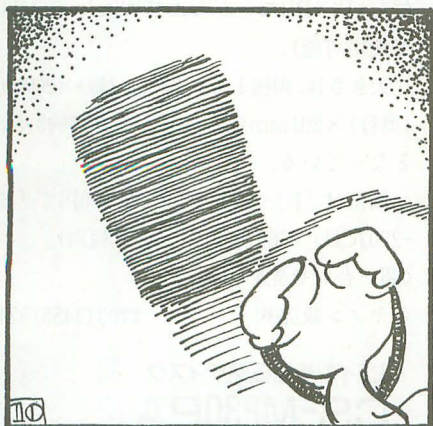
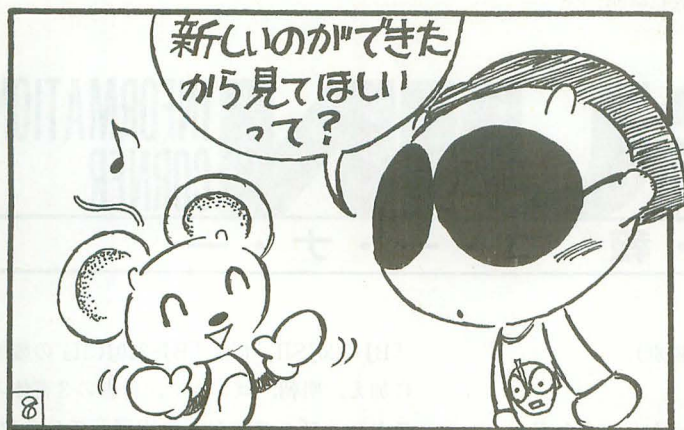
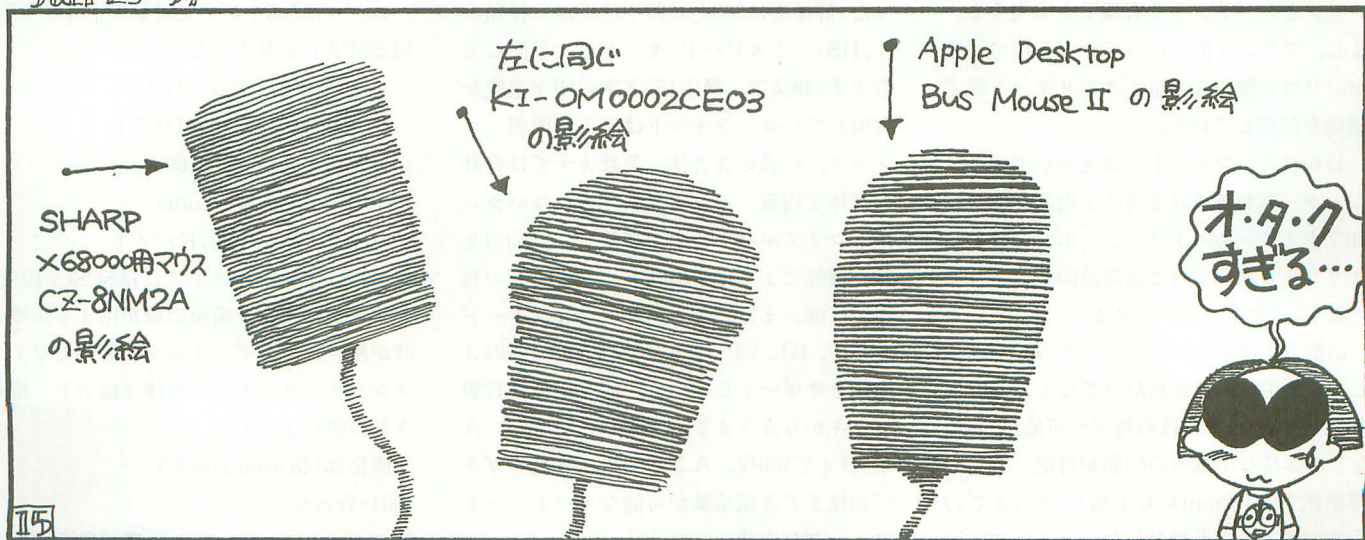
新しいのができたから見てほしいって？

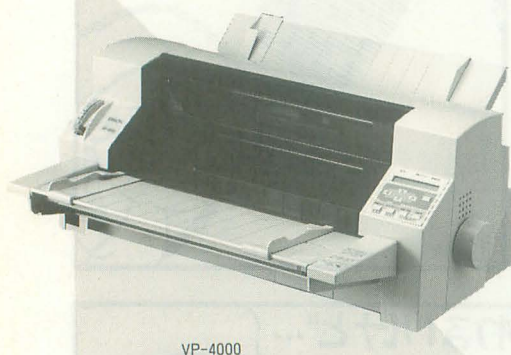
どれどれ…

わ…わるいけど…
全然わかんない…

JULY 25 '94

SHARP
X68000用マウス
CZ-8NM2A
の影絵
左に同じ
KI-OM0002CE03
の影絵
Apple Desktop
Bus Mouse II の影絵
オ・タ・ク
すぎる…

PENGUIN INFORMATION CORNER
ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

プリンタ VP-4000 セイコーエプソン

VP-4000

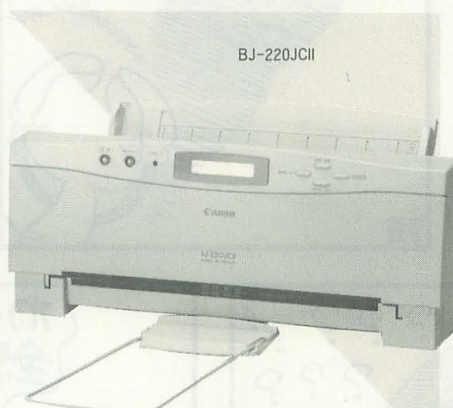
セイコーエプソンは136桁対応のインパクトプリンタ「VP-4000」を発売した。

本機はネットワーク環境を重視した機能が装備されている。まず，拡張スロットにオプションのネットワーク用インタフェイスカードを装着することにより，ネットワーク環境に対応し，パソコンによるプリンタサーバの必要がない。そして，最大3種類(オプションインタフェイス装着時)のパソコンとのプリンタ共有環境を実現する。また，プリンタ側で行っていた各種の設定をパソコン側から設定できるリモート設定機能を装備している。

ほかにも，フォントには漢字が明朝とゴシック，英数カナは2書体を内蔵。カラー印字もカラーカートリッジリボンの装着により行える。印字速度は高速印字で1秒間に漢字160文字の印字が可能。

給紙方式は水平ローディング方式を採用し，単票用紙や連続紙だけでなく，複写伝票(オリジナル＋5枚の複写が可能)や封筒などの多様な用紙への印刷が可能。また，単票紙上端4.2mmから下端4.2mmまでの印字ができるようになった。

価格は228,000円(税別)。
〈問い合わせ先〉
エプソンインフォメーションセンター
☎0424(99)7133，06(399)1115

プリンタ BJ-220JCII/JSII キヤノン

BJ-220JCII

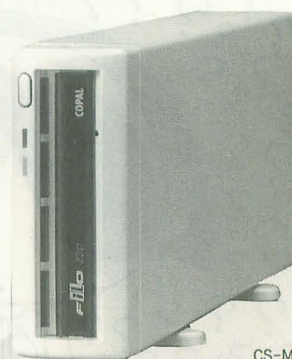
キヤノンはバブルジェットプリンタ「BJ-220JCII」「BJ-220JSII」を発売した。

「BJ-220JCII」は写真やイラストなどの連続階調の画像を出力するときに効果的なグレイスケールに対応している。スムージング機能により，ビットマップフォントでも4倍角の文字のなめらかな印字が可能になった。解像度は360dpi。印字速度は3種類あり，HS(ハイスピード)モードで1秒間に英数カナ248文字，漢字165文字の印字速度を実現している。フォントは漢字が明朝，ゴシック，行書の3書体，英数カナでは合計6書体を内蔵。インタフェイスにはパラレルとシリアルを1つずつ搭載し，自動切り替え機能によって2種類のパソコンでの利用が可能。また，エミュレーションモードとして，BJ，VP-1700，PC-PRモードの3種類をサポートしている。印刷用紙は官製ハガキからA3までの大きさに対応し，A4/B4で100枚，A3で50枚，官製ハガキで40枚まで連続給紙が可能なオートシートフィーダを内蔵。

「BJ-220JSII」では「BJ-220JCII」の機能に加え，明朝，ゴシック，行書の3書体のスケーラブルフォントを内蔵している(ただし，PC-PRモードおよびWindows 3.1にて使用可能)。

大きさは，両機ともに428mm(幅)×209mm(奥行)×201mm(高さ)で，重さが約3.5kgとなっている。

価格は「BJ-220JSII」が89,800円で「BJ-220JCII」が69,800円(ともに税別)。
〈問い合わせ先〉
キヤノン販売㈱ ☎03(3455)9544

3.5インチ光磁気ディスク CS-M230PA コパル

CS-M230PA

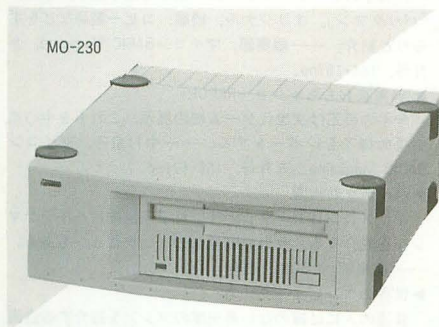
コパルは3.5インチ光磁気ディスク「CS-M230PA」を発売した。

主な仕様は以下のとおり。

記憶容量：128/230Mバイト
ディスク回転数：3,600rpm
平均シーク速度：35ms
バッファ容量：237Kバイト

また，インタフェイスにはSCSIを採用。ディスク挿入時も前面の扉が閉まる防塵設計が施され，エアフィルタの必要もなく，メンテナンスは不要。本体は縦置き，横置きが自由に選べる。

価格は148,000円(税別)。
〈問い合わせ先〉
㈱コパル ☎03(3558)3582

3.5インチ光磁気ディスク
MO-230
ユニバーサル

MO-230

ユニバーサルは3.5インチ光磁気ディスク「MO-230」を発売した。

主な仕様は以下のとおり。

記憶容量：128/230Mバイト

ディスク回転数：4,200rpm

平均シーク速度：28ms
(230Mバイト使用時)

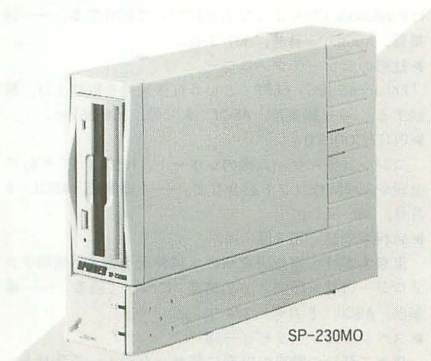
バッファ容量：256Kバイト
(オプションで1Mバイト対応可)

また，インタフェイスはSCSIを採用。レンズクリーニングやエアフィルタなどのメンテナンスは不要。

価格は138,000円(税別)。

〈問い合わせ先〉

ユニバーサル㈱　☎06(354)7301

3.5インチ光磁気ディスク
SP-230MO
ジェフ

SP-230MO

ジェフは3.5インチ光磁気ディスク「SP-230MO」を発売した。

主な仕様は以下のとおり。

記憶容量：128/230Mバイト

ディスク回転数：3,600rpm

平均シーク速度：35ms

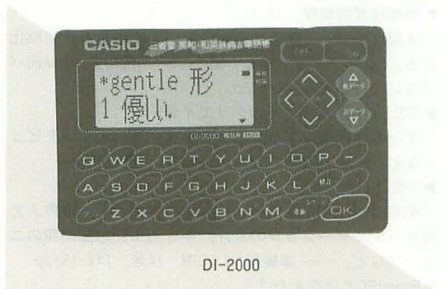
バッファ容量：256Kバイト

また，インタフェイスにはSCSIを採用している。メンテナンスはエアフィルタを必要としない防塵設計により不要。

価格は148,000円(税別)。

〈問い合わせ先〉

㈱ジェフ　☎06(336)5901

英和・和英電子辞書
DI-2000
カシオ計算機

DI-2000

カシオ計算機はカードサイズの英和・和英電子辞書「DI-2000」を発売した。

同機は英和辞典機能としてニューセンチュリー英和辞典から約47,500語を収録。和英辞典機能は，新クラウン和英辞典から約31,000語を収録している。使用方法は訳したい単語を入力して，OKボタンを押すだけで単語とその訳を表示してくれる。

ほかにも，150人分の電話番号が記憶できる漢字電話帳機能や電話番号などのデータをパスワードで守るシークレット機能，独立メモリつき10桁計算機能などが装備されている。

価格は12,000円(税別)。

〈問い合わせ先〉

カシオ計算機㈱　☎03(3347)4811

ラベル印刷機
KL-800/KL-810
カシオ計算機

KL-800

カシオ計算機は家庭用ラベル印刷機2機種「KL-800」「KL-810」を発売した。

同機はビデオカセットのタイトルラベルなどを画面の指示に従って作成していく。主な特長としては，家庭でよく使うと思われるラベルのフォーマット38種類を内蔵。印字書体は見やすさを考慮してゴシック体を採用。文字以外では，フレーム印刷機能により，ラベルの周囲に多彩な装飾が可能。また，モンタージュの要領で似顔絵を作成し，ラベルに印刷することができ，最大20個まで登録できる。

ほかにも，レイアウト表示機能により，印刷イメージの確認ができる。ジャストフィット印刷機能により，行数に合わせて文字サイズや印刷モード(標準/縮小)を自動的に設定して印刷を行ってくれる。印字解像度は200dpiで，印字するためのテープカートリッジは74種類用意されている。

「KL-800」はキーの配列が50音順で，「KL-810」はJIS準拠になっている。

価格はどちらも18,000円(税別)。

〈問い合わせ先〉

カシオ計算機㈱　☎03(3347)4811

BOOK

最新MIDI DATA制作術
基礎編/実践編
東亜音楽社

東亜音楽社は『最新MIDI DATA制作術 基礎編』『最新MIDI DATA制作術 実践編』を発売した。

同書はMIDIデータを制作するうえでのポイントをSC-55とレコンポーザ(PC-9801用)を使って解説していく。

基礎編では音楽の基礎から耳コピーの方法，ミキシング，エフェクトなどについてふれている。

実践編は，リズムパターンや入力のテクニック，コード，アレンジ技法など。

価格はどちらも3,000円(税込)。

〈問い合わせ先〉

㈱東亜音楽社　☎03(3260)6271

FILES Oh!X

このインデックスは，タイトル，注記――著者名，誌名，月号，ページで構成されています。照りつける日差しが眩しい今日この頃，暦のうえでは秋ですが，まだまだ暑さは続きそうです。体のために もしっかり睡眠をとりましょう。

参考文献
I/O　工学社
ASAHIパソコン
ASCII　アスキー　朝日新聞社
コンプティーク　角川書店
C MAGAZINE　ソフトバンク
電撃王　主婦の友社
PIXEL　図形処理情報センター
マイコンBASIC Magazine　電波新聞社
My Computer Magazine　電波新聞社
LOGIN　アスキー

一般

▶NEWS
「WINDOWS WORLD Expo/Tokyo」の模様や，相次ぐパソコン通信訴訟の話題など。――編集部，ASAHIパソコン，7・15号，8-11pp.

▶NEWS&VIEWS
ピーター・ガブリエルの「エクスプローラ1」の内容から，音楽とCD-ROMビジネスの可能性を探る。――鍛治信太郎，ASAHIパソコン，7・15号，12-15pp.

▶便利な小道具で四角いパソコンを丸く使う
パソコンの周辺にある便利なグッズを紹介する。ディスプレイフィルタや変わりダネマウスなど。――西田宗千佳，ASAHIパソコン，7・15号，116-121pp.

▶機械用言博物館　13
今回はメディア関連の用語について解説する。「初期化する」「フォーマットする」など。――荻窪圭，ASAHIパソコン，7・15号，124-125pp.

▶GO! GO! 未来のゲームクリエイター
ゲーム作家養成学校の紹介や現役生徒へのインタビューなど。――編集部，LOGIN，14号，123-143pp.

▶THE NEWS FILE
「WINDOWS WORLD Expo/Tokyo」のレポート，音声入力型カーナビゲーションの紹介，手塚治虫記念館開館のニュースなど。――編集部，LOGIN，14号，144-151pp.

▶PowerPCとはなんだ？
RISCの始まりや「PowerPC」とはどんなものかを解説する。――編集部，LOGIN，14号，182-185pp.

▶架想楽園へ行こう　Ver2.02
次世代ゲーム機のメーカーがなにを考え，どういう戦略をもって取り組んでいるのかをチェックする。――中田宏之，LOGIN，14号，186-189pp.

▶くねくね科学探検
鹿野司が人間と環境のインタフェイスについて語る。――鹿野司，LOGIN，14号，196-197pp.

▶モービル・コンピューティングとマルチメディア
「ビジネスショウ'94」の模様をレポート。携帯情報端末とマルチメディアマシンにスポットを当てる。――編集部，コンプティーク，8月号，114-117pp.

▶'94東京おもちゃショー 徹底紹介!!
「'94東京おもちゃショー」のレポート。次世代ゲーム機を中心に気になる最新のおもちゃを紹介する。――編集部，コンプティーク，8月号，124-129pp.

▶NEWS COLLECTORS
アメリカで来年春に発売と噂される任天堂32ビット機，コードネーム(?)「VR」に関する情報など。――電撃王，8月号，20-23pp.

▶任天堂次世代機ついに正式発表！
任天堂の次世代ゲーム機「ULTRA64」の情報を紹介。「CES」などのレポート，各社の次世代ゲーム機に関する最新情報。――編集部，電撃王，8月号，26-41pp.

▶DENGEKI SUPER HIT CHART
コンシューマ，パソコン，アーケードのゲームの販売動向をデータをもとに分析。推定売上本数まではじきだす。――編集部，電撃王，8月号，96-103pp.

▶木製ジェットコースターのひ・み・つ
よみうりランドの「ホワイトキャニオン」の取材をとおして木製ジェットコースターの秘密を探る。――編集部，電撃王，8月号，132-135pp.

▶真説武術体系
実在の武術を取り上げて，その技術や歴史，達人の逸話などを聞く。語り手はコミック「拳児」の原作者，松田隆智氏。――編集部，電撃王，8月号，144-145pp.

▶マルチメディア・パソコンでテレビ＆ビデオを観る！
TVチューナー内蔵パソコンを機種ごとに紹介する。また，「ビデオPC for X680x0」「ReelMagic」などのビデオCDを観るための製品も紹介する。――編集部，マイコンBASIC Magazine，8月号，35-46pp.

▶CD-ROMからはじめるマルチメディア　第3回
今回のテーマは，3倍速と2倍速のドライブの差について。動作時間の測定をもとに，3倍速ドライブのメリットはどこにあるかを考える。――吉岡哲也，マイコンBASIC Magazine，8月号，56-57pp.

▶新ハード情報局
次世代ゲーム機の現在のソフト開発状況をレポートする。――山下章，マイコンBASIC Magazine，8月号，147-150pp.

▶Arcade Game Graffiti　第6回
1980年のアーケードゲームの動きを振り返る。今回は「パックマン」。オリジナル，続編，コピー製品などをずらりと紹介。――編集部，マイコンBASIC Magazine，8月号，164-167pp.

▶'94東京おもちゃショーREPORT
今年の目玉は次世代ゲーム機の展示。これらを中心に，会場の様子をレポートする。――中村京子，マイコンBASIC Magazine，8月号，168-171pp.

▶ハイテク幼稚園
「キッズコンピュータ」と呼ばれる子供向けハイテクマシンを紹介する。セガの開発者のインタビューもある。――編集部，LOGIN，15号，182-185pp.

▶世界のコンピューター君
普通の人には縁のない最先端のマシンを紹介する企画。今回は，「ジュラシック・パーク」で活躍したグラフィックワークステーション，INDYの素顔に迫る。――編集部，LOGIN，15号，200-203pp.

▶くねくね科学探検
木星への巨大彗星激突。この彗星と日本の天文界について紹介する。――鹿野司，LOGIN，15号，204-205pp.

▶NEWS&VIEWS
ワールドカップサッカーの円滑な運営を支える情報システムについて取材する。――長原匡史，ASAHIパソコン，8・1号，12-17pp.

▶カラー・インクジェットプリンタ3機種5番勝負
「BJC-600J」「DeskJet560J」「MJ-700V2C」の3機種を速度や画質，機能など5つの視点から比較する。――斉藤幾郎など，ASAHIパソコン，8・1号，20-31pp.

▶機械用言博物館　14
「開く」という用語を取り上げ，主にデータのロード，セーブに関する説明を行う。――荻窪圭，ASAHIパソコン，8・1号，128-129pp.

▶最新パソコン・ガイド
過去のPC-9801シリーズと最新機種とで処理能力を比較する。ほかに「PS/V Vision」「Performa575」などの新機種を紹介。――編集部，I/O，8月号，27-42pp.

▶MultiMedia Watching　8
今回はマルチメディアターミナルによる通信を考える。――奥野雅之，I/O，8月号，129-132pp.

▶特集I　夏の新製品戦線
目的別，マシンの性格別に新製品をまとめて紹介する。――編集部，ASCII，8月号，261-292pp.

▶特集II　"Chicago"のこと全部教えますPart1
「Chicago」について得られた情報を逐次レポート。今回はWindows3.1からよくなる点について紹介する。――編集部，ASCII，8月号，301-316pp.

▶魅惑のニューテクノロジー
「Enhanced IDE」「EPP」という拡張規格を取り上げ，解説する。――編集部，ASCII，8月号，354-359pp.

▶INTERCOOLED
コンシューマゲーム機のレポート。NECの「PC-FX」の全貌や3DO新作ソフト紹介など。――編集部，ASCII，8月号，366-369pp.

▶新科学対話　第8回
東京大学社会情報研究所の水越伸氏を迎え，「情報テクノロジーの将来予測とその結果」について語る。――編集部，ASCII，8月号，374-380pp.

▶スペシャルインタビュー16
日本で最初の電卓の開発に関わったシャープ副社長，浅田篤氏にそのエピソードを聞く。――遠藤諭，ASCII，8月号，404-409pp.

▶稀代ものの け考
海外のおかしなグッズを紹介する。オーディオお掃除用バキュームノズル，潜水艦ラジコンなど。――バカパパ，ASCII，8月号，438-439pp.

▶特集　プリンタ新製品カラー・スピード徹底比較
プリンタの選択のポイントや新機種のパフォーマンスを比較する。――編集部，My Computer Magazine，8月号，7-20pp.

▶光磁気ディスクの勧め(1)
MOの全般的な概論について説明する。メディアの特長など。――佐田守弘, My Computer Magazine, 8月号, 59-63pp.
▶パソコン研究室
MOディスクの仕組み，構造と使い方を解説する。――編集部, My Computer Magazine, 8月号, 135-139pp.
▶ビジネスマンのための情報管理術
「PI-4000」「PI-4000FX」を取り上げ，インクワープロ機能，FAXによる送受信などの新機能を解説。――塚田洋一, My Computer Magazine, 8月号, 140-143pp.

MZシリーズ

MZ-2500(BASIC-M25)
▶BW
砂漠にまき散らしたフロッピーを回収する。――アダモ, マイコンBASIC Magazine, 8月号, 89-91pp.

X1turbo/z

X1シリーズ
▶トトカルチョFIGHTERS
パンチとキックで戦うキャラクター格闘対戦ゲーム。――石井一鑑, マイコンBASIC Magazine, 8月号, 109-110pp.

X68000

▶NEWSOFT
TAKERU名作文庫ソフトシリーズ第2弾を紹介。「三國志II」などがX68000用に続々登場。ほかに新作ソフト情報。――編集部, LOGIN, 14号, 6-25pp.
▶X68新聞
「餓狼伝説SPECIAL」の情報，「ビデオPC for X680x0」の紹介など。今回が最終回。――編集部, LOGIN, 14号, 164-165pp.
▶S.S.express
「餓狼伝説SPECIAL」や「雀神クエスト」などX68000を含めた各機種用の新作を紹介。――編集部, コンプティーク, 8月号, 23-44pp.
▶How To Win
ゲーム内容や攻略法を紹介。X68000用には「スーパーリアル麻雀PIV」など。――編集部, コンプティーク, 8月号, 51-89pp.
▶電撃王全ゲームインデックス
各機種のゲームデータと情報コーナー。X68000版「宝魔ハンターライム最終回」など。――編集部, 電撃王, 8月号, 6-7pp.
▶新作王
8月以降に発売されるゲームを中心とした情報コーナー。X68000版「レッスルエンジェルス3」など。――編集部, 電撃王, 8月号, 167-185pp.
▶SCROPE
ロープを使ったスクロールアクションゲーム。――仁井内明, マイコンBASIC Magazine, 8月号, 111-113pp.
▶勝利の祝砲
2人で戦うタンクバトルゲーム。――酒井健児, マイコンBASIC Magazine, 8月号, 114-116pp.
▶TYQ
あなたの不得意なキーを集中的に訓練する，タイプ練習プログラム。――西井利明, マイコンBASIC Magazine, 8月号, 117-119pp.
▶同人野郎
同人ソフトを紹介する。通販申込先つき。X68000はシューティングゲーム「XADLAK」ほか4作。――安部理一郎, マイコンBASIC Magazine, 8月号, 194-195pp.
▶SUPER SOFT HOT INFORMATION
X68000用「スーパーストII」移植決定のニュースなど。――編集部, マイコンBASIC Magazine, 8月号, 別冊1-39pp.
▶NEWSOFT
「餓狼伝説スペシャル」などの新作ゲームソフト情報。――編集部, LOGIN, 15号, 10-25pp.
▶ONLINE SOFTWARE INDEX
大手ネットにアップロードされたソフトを紹介する。X68000用「SXCDP.X」「FSXPATCH.X」など。――編集部, ASCII, 8月号, 482-487pp.
▶なんでもQ&A
「MUSIC SX-68K」で斜めビームを入力する方法などX68000に関する質問に答える。――編集部, My Computer Magazine, 8月号, 181-182pp.
▶SX-WINDOWプログラミング　第10回
今回は「SX-WINDOW ver.3.1」と「SX-WINDOW開発キット用ツール集」の試用レポートを行う。――吉野智興, C MAGAZINE, 8月号, 129-134pp.
▶シャープXグループ
「SX-WINDOW ver 3.1」に付属するシャーペンの外部コマンドについて解説する。――編集部, C MAGAZINE, 8月号, 152p.

ポケコン

PC-E500
▶クイズケイバ
5人の解答者のなかから誰が正解するかを当てるクイズゲーム。――命かけます授業中の兄, マイコンBASIC Magazine, 8月号, 120-121pp.

新刊書案内

電視遊戯時代
テレビゲームの現在
テレビゲーム・ミュージアム・プロジェクト編
ビレッジセンター出版局刊
☎03(3221)3520
B5判　207ページ
4,980円（税込）

1988年にUPUから出た「電視遊戯大全」というでかい本があった。本書はその続編だと考えていいだろう。しかし，1冊の本としてちゃんと独立しているし，現在の技術，視点から電視遊戯を見直してみた，という観点で捉えれば，「電視遊戯大全」をリプレースするものと考えてもよい。それが「電視遊戯時代」である。
副題は「テレビゲームの現在」で，これはあらゆる角度からテレビゲームを追究した本。我々は黎明期からテレビゲームで仮想世界を楽しんできたのだ，いまさらマルチメディアなんて企業の人たちが大騒ぎしたってなんてことないのさ，という感じ。そういった，テレビゲーム黎明期から現在まで，そのまっただなかで揉まれ，育ってきた人たちによる，「そろそろ一度テレビゲームについて整理してみようか」という本なのである。原則として見開き1単位とし，テレビゲームの歴史，1993年の現状（このパソコンゲームのページはかなり甘い），テレビゲームからマルチメディアへ移るとすればどんな流れになるのか，テーマ別に探ってみるエッセイ，テレビゲーム論やインタビュー，そして，「電視遊戯大賞1994」で構成されている。見開き単位だからパラパラと眺めてもいいし，図版も豊富だから，資料的価値もある。ネタの幅も広く，エッセイの詰まった第3章では，さまざまな人たちが，情報ハイウェイ，ゲームにおけるスターキャラクターの意味，メガデモ，「シムもの」の功罪，ゲームミニコミなどなどについて述べている。
高い本だけれども，カラフルだし，レイアウトも凝っているし，なかなかよい。ただ，パソコンゲームについての言及が甘い点が気になる。家庭用ゲーム機のゲームの多くはパソコンゲームに源流を求めているのだから。
ちなみに，電視遊戯大賞1994は「バーチャファイター」である。　（K）

最新パソコン技術大系
日経バイト編
日経BP社刊
四六判　408ページ
3,800円（税込）

パソコンの世界は日進月歩，いや秒進分歩ともいわれる。そうするといままで耳なれなかった言葉が雑誌のなかにもあふれてくる。
本書ではパソコンの各種専門用語について解説していく。内容は，ハードウェア，周辺装置，OS，LAN，資料の5部構成で，それぞれいくつかの章に分けて，より詳しく述べている。
パソコン雑誌を読んでもわからないことが多かったり，現在のパソコン技術がどんなレベルにあるのか知りたい人は読んでみるといいだろう。ただ，いまこの瞬間にも新しい技術は生まれているのだが。

実戦的お天気入門
阿施光南著
山海堂刊
☎03(3816)1617
四六判　190ページ
1,800円（税込）

誰しも，天気予報を試みたことがあるだろう。ただ，「高気圧がこのあたりにあるから明日は晴れかな？」この程度ではないだろうか。では，なぜ高気圧があると晴れるのだろうか？
本書ではいろいろな角度から天気を見つめ，わかりやすく解説していく。主な内容は，天気のメカニズムに関する基礎知識から気象情報を活用するための常識，経験と科学の常識，アウトドアの体験などから学ぶ気象変化の読み方などである。
パソコンを離れてたまには外に出よう，と思ったときに雨ではちょっとつまらない。そんなことにならないために少し勉強してみては。

QUESTION and A

6月号47ページの「インタレースの謎」の最初に「SX-WINDOWの隠しオプション-Gを……」という一文があったので素直に,

SXWIN -G

を実行したところ,画面が512×512モードになってしまいました。Gの後ろにそれらしい数字をつけてみたのですが,元に戻りません。画面を初期化する方法を教えてください。　北海道　加藤　英輝

SXWIN.Xの起動オプションGはSX-WINDOWで使用する画面モードを指定するためのものです。

ここで指定されたパラメータはSX-WINDOW起動時にSRAMに登録されますので,次からオプションなしで起動したときにはそのとき指定した値が参照されるようになります。ですから間違った値を与えると少しやっかいなことになるのです。

-Gで指定する値はIOCSコールCRTMODに与えるパラメータを基本としており,概ね表1のような仕様になっています。この基本数値に32を加えると,SX-WINDOWの実画面モードとなり画面スクロールができるようになります。

ですから一般的な使用状態の場合は,

SXWIN -G16

ないしは,

SXWIN -G48

で一度起動してから終了し,次回からは単に,

SXWIN

で立ち上げるようにしてください。

6月号の記事で指していたのは-G18あたりのことですが。24kHzインタレースモードを使用したモードなどが指定できますが,ちらつきも大きいので最近は使う人もいないみたいです。

表1　SXWIN -Gのパラメータ

Gの値	ドット数	色数	周波数
0	512×512	16	31kHz
1	512×512	16	15kHz
2	256×256	16	31kHz
3	256×256	16	15kHz
4	512×512	16	31kHz
5	512×512	16	15kHz
6	256×256	16	31kHz
7	256×256	16	15kHz
8	512×512	256	31kHz
9	512×512	256	15kHz
10	256×256	256	31kHz
11	256×256	256	15kHz
12	512×512	65536	31kHz
13	512×512	65536	15kHz
14	256×256	65536	31kHz
15	256×256	65536	15kHz
16	768×512	16	31kHz
17	1024×424	16	24kHz
18	1024×848	16	24kHz

それぞれ+32で実画面モードとなる

8月号に石上達也氏のアクセラレータのベンチマークテスト結果が載っていましたが,私が考えるに,68030は68000に比べて命令が2,3倍速かったと思います。クロックが20MHzと2倍になっているのだからメモリアクセスが16ビットで遅くなっているとしても,キャッシュなしで4倍程度の速度が出るような気がしますが,ベンチマークテストでは数%しか速くなっていません。どうしてでしょうか。　滋賀県　西尾　幸造

レジスタ間の演算だけならともかく,16ビットバス時の68030では所定の性能は発揮できません。

68030のマニュアルで命令実行速度関係の表を見ると,そこには「Iキャッシュケース」「ノーキャッシュケース」という表記があります。これは「命令キャッシュがヒットするかどうか」ということですが,キャッシュ動作しているかどうかは命令自体の実行時間にも多少の影響を与えます。さらにキャッシュはメモリウエイトの回避につながりますから,当然プログラムの実行時間も変わってきます。

ちなみに,今月号のX68030D'ashの記事中のベンチマークテストではデータキャッシュが動作するバージョンの結果が記載されていますので参考にしてください(まだ遅いけど)。

さらに但し書きを見ると,すべての項目について逐一「すべての実行時間データは2クロックのリードおよびライトを想定しています」という表記があります。68030では最短クロックサイクルは同期バス時に2クロック,非同期バス時に3クロックです。X68030は同期バスでメモリが接続されていますが,このアクセラレータではもちろん非同期バス接続です。ほかにも「すべてのオペランドはロングワードに整列している」とか「すべてのバスサイクルに対して必要な変換はアドレス変換キャッシュに存在している」という条件が示されています。

肝心の命令の実行時間自体ですが,68030のマニュアルでは実行時間と命令フェッチ時間とアドレス計算時間は別にまとめてあったり,68000のマニュアルとは少し見方も違ってきますので気をつけてください。

68000関連の機械語命令でもっとも多用されるであろうMOVE命令で見ると,レジスタから(An)への転送は3クロックで,うち1クロックは削られる可能性があります(キャッシュは影響しない)。68000の場合はワードサイズ時に8クロック,ロングワード時に12クロックです。

逆に(An)からデータレジスタへの転送だと,68030の場合,フェッチ3クロック+実行2クロックの5クロックで,可能性は低いのですが1クロック削られることもあります。68000は先ほどと同じクロック数です。こういったものを見ると命令レベルで2～4倍というのは必ずしも間違いではありませんが,これも32ビットバス時の値です。16ビットバス接続だと16ビットずつ2回に分けてアクセスして32ビットデータを作りますので1メモリアクセスサイクル(非同期バス時3クロック)以上の遅れが加算されてきます。

プログラムやキャッシュの効き具合などにもよりますが,一般に,68030が16ビットバスで接続された場合,同クロックの68000と同程度というのが定説になっています。

8月号の時点のボードではデータキャッシュは使用できないので,この点ではかなり不利になっています。加えて,本文中にも説明があったとおり,DTACK先出しの不備で1クロック,その他の処理でもう1クロック近く余分なウエイトが入るようになっています。

では68030にするメリットはというと,まず,高クロックに対応しているということ,コプロセッサが接続できるということにあります(乗除算の拡張やシフトの高速化といった命令レベルでのメリットも多少はありますが)。

ちなみに,DTACKの先出しを禁止した状態でのX68000ノーマル機の数値も書いてありましたのでそれと比べてみましょう。すると20MHzキャッシュなしだと約33%速くなっていることがわかります。キャッシュが効けば16ビットバスのためにボトル

ネックになっているメモリアクセスが減ることになり，格段に速度は変わってくるはずですので，この33%という数値は最悪時のパフォーマンスを示していると見てもいいでしょう（普通はキャッシュを入れて使う）。

石上版アクセラレータは，コスト的に見て30MHz動作が妥当ではないかというふうに進めていますので，最終的にX68000ノーマル機の3，4倍程度の速度になると思われます。

一般に32ビットマシンの命令実行時間は素直には決まりません。ウエイトやパイプラインの流れによって変動してくるからです。マニュアルに記載してあるものは，32ビット同期バス接続でキャッシュが当たったときを規準にしています。メモリウエイト時のパフォーマンスなどは別に複雑な計算法が示されていますが，ウエイトがひとつ入るだけでもかなりパフォーマンスは低下します。いずれにせよ，32ビットバス接続されていないと所定の性能は発揮できないのです。

以前，1991年1月号の付録ディスク「謹賀新年PRO-68K」のなかに入っていたZ's-EXなんですけど，やっぱり私も「未登録の割り込み」となって動作しませんでした。Z'sSTAFFはジョイスティックのプロテクタが付いたやつなのですが，どうか動作させる方法があったら教えてください。Human68kはver.2.01，FLOATもver.2とか付録のとかを試してみました。　群馬県　関口　孝紀

ソフトはできるだけ最新版を使ってほしいところですが，とりあえず，可能性のありそうなところから解説してみましょう。

関口さんがお持ちのものは，プロテクトモジュールつきのZ'sSTAFFというと，これはver.1.0の初期版ということになりますね。

Z's-EXをZ'sSTAFFのver.1で動かす場合には，あらかじめグラフィック画面が初期化されている必要があります。これを行わない場合には「未登録の割り込み」が出てくるはずです。そこでまず，

SCREEN 1 3 1

を実行したうえで，Z's-EXを動かしてください。

Z'sSTAFFではなんらかのシステムエラーが発生した場合はすべて「未登録の割り込み」として表示されるので，これ以外の理由で起動に失敗しているときには十分な対処ができません。Z's-EXのver.2以降では多少の対策もしてありますので，とりあえず，メモリを十分に空けて試してみてそれでも駄目だったら新しいバージョンに換えてください。

8月号の「XsimmVI」の紹介記事で「Compactユーザーで数値演算プロセッサが使えないのが痛いかもしれない」と書いてありましたが，これはどういうことですか。もしかしてCompact以外は数値演算プロセッサが使えなくとも痛くないということですか。　宮城県　柳田　正幸

これはX68000XVIとX68000 CompactXVIの増設RAMボードの仕様が違うことに起因する問題のことをいっているのですが，少し表現が悪かったようです。

もともとのX68000XVIの内蔵用増設メモリボードにはメモリ2Mバイト，増設メモリコネクタ×2，IOCS用ROMソケットが装備されていますが，X68000CompactXVI用のものでは，メモリ2Mバイト，増設メモリコネクタ×2，IOCS用ROMソケット，数値演算プロセッサソケットが装備されています。

要するに，X68000XVIでは本体基板上に設置されていた数値演算プロセッサソケットが増設メモリボード上に載っかったということです。

この純正メモリボードというのは，実は機能としては単なるメモリボードではなかったわけですが，XsimmVIは純粋なメモリボードとなっています。よって，X68000 CompactXVIでXsimmVIを使うと数値演算プロセッサを内蔵できない，ということになるわけです。

数値演算プロセッサを必要とする，あるいはないとつらい処理というのは限られていますので（3Dグラフィックの演算とか），これまで不便でなかった人にはあまり必要はないものでしょう。数値演算プロセッサを積んでも通常の処理はほとんど速くなりません。

もちろんこの状態でもスロット用の数値演算プロセッサボードを使用できますので，どうしてもという方は外につけることはできます。内蔵ソケットが用意されているX68000XVIで外づけの基板を使っても問題はなにもありません（内蔵だと0.66MHz分速く動くということくらいですか）。

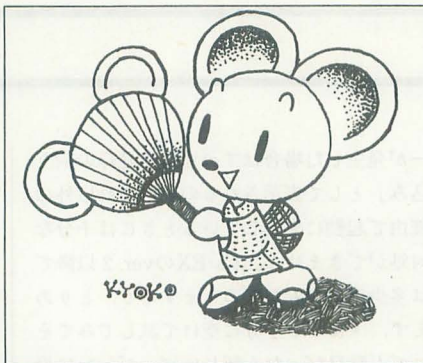
ちなみに，IOCS用のROMソケットというのはIOCSをバージョンアップする際に，この空きソケットにROMを入れてジャンパスイッチで切り換えると新しいROMで起動するマシンになるというものです。X68000シリーズではすべての機種でこのような処置が施されています（初代機にはROMソケットはついてないが元のROMがソケットについていた）。

このROMソケットは，拡張性やバグ回避を保証するものですが，これまでの例からして致命的な問題点も出ていませんし，IOCS関係の拡張はソフトで行われる方針になっているようですから，さしせまって必要になることはないでしょう。ユーザーレベルで見ても，少なくとも私の知る限りではまだ使われたという話を聞いたことはありません。ただし，今後なにがあっても不思議はないような昨今の状況ではありますが……。

最後にX68000シリーズではCPUまわりを除いてはハード，ソフトともにほとんど同じ仕様になっていますから（ディスクドライブのタイプは別として），CompactXVIだからなにかの機能が足りないというようなことはありません。（中野　修一）

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を挙げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに解答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので電話番号も明記してください。

宛先：〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク株式会社出版部
Oh!X編集部「Oh!X質問箱」係

FROM READERS TO THE EDITOR

そろそろ暑さも一段落。熱くなった肌も冷め，初秋の香りもただよう頃。夏のあわただしさも落ち着きを取り戻し，いろんなことに集中しやすくなりました。学生の方は夏休みボケを治しておかないと，あとで泣きをみますよ(経験者談)。

◆「GEINIE」はvery goodでした。「DōGAの皆さん，ありがとう」と伝えてください。最近はSX-WINDOWに凝っていて，CGAはあまりやらなかったのですが，またやる気になりました。

竹内　孝雄(31)大阪府

◆な，なんなんだ7月号の付録ディスクは……。こんなものを付録につけていいの？　スゴイですね。これだけで，本誌数カ月分の価値がありそうです。Oh!Xを買っていてよかったと心から思いました。　深見　満彦(17)和歌山県

◆こんな付録はTAKERUにでもまかせてほしい。これで＋250円は痛い。そこまでして無理やり付録ディスクをつけることはないでしょう。

高橋　毅(23)埼玉県

7月号の付録ディスクにも賛否両論ありましたが，ほとんどの方が喜んでくれたようです。いままではとっつきにくかったけれども，「GENIE」の扱いやすさに感激したという方が多かったようです。これをきっかけにCGAにがんばって挑戦してみてください。

◆楽譜の情報量というのは，あまり多くない。その分を演奏者が独自の解釈(センス)で補うのである。だから，初心者にも使える楽譜入力ソフトだと，演奏のクオリティはいまいちになるだろうし，いろいろな情報も書き込めるようにするくらいなら，MMLを使ったほうがよいと思う。あとは，楽譜の読める(ちゃんと解釈のできる)人工知能を開発するとか(ゲゲッ)。でもそれだと人工知能のセンスで演奏しているだけで，入力した人のセンスは反映されない。やはりMMLかコンピュータを使わない音楽をやるのがよいだろう。　西尾　昌人(20)愛知県

楽譜入力だけでは細かなニュアンスは再現できないかもしれません。でも，多くの人がコンピュータミュージックに親しむためには，あったほうがよいのでは。

◆以前に発表された電車の発車のメロディといい，中央競馬のファンファーレといい，かなり以前から思いついていたのに……。やはり，思いついたらすぐに作るというのが基本ですね。もう1曲考えているのがあるのですが，完成するかな？　北村　満(24)神奈川県

急がないと先を越されてしまうかも。

◆響子さんの実用講座はPhotoCDってこんな使い方ができるのかと感心させられた。でも，このためにCD-ROMドライブを買うのは無理で残念。これからも，このようなパソコンの使用法を紹介してほしい。　美辺　央希(20)東京都

CD-ROMの使い方はほかにもあるので，これを機会に買ってみてはいかがでしょうか。

◆「データがない！」。いままでは，徳川埋蔵金よろしく，発掘していたのですが，ついに発見不可能のものが出てしまいました。やっぱり，整理は大事ですね。「猫とコンピュータ」を読んで，つくづく思いました。

猿渡　哲治(22)福岡県

整理は大事だと思いつつも，机の周囲を見回すと……。

◆暑い，暑い，暑いー。X68000の動作温度は10℃～35℃と書いてあるけど僕の部屋は，AM11時現在，34℃(ちなみに安物の温度計なので34℃までしか計れない)ときたもんです。壊れるんじゃないかと毎日ヒヤヒヤしています。だいたい僕の部屋は2月ぐらいから，クモ，ゴキブリ，アリ，ムカデ，ヤモリが出てくるほど暖かいのです。風通しはいいハズなのだが……。

廣田　祥巳(19)千葉県

社内では，冬よりも夏のほうが寒いような気がします。パソコンにとっては悪くないのですが，人間は……。

◆知人から，この度ハードディスクを譲りうけまして，「これからは下の欄に記入できるわ」と喜んでおります。彼いわく「いまどきハードディスクすら持ってないなんて，かわいそうすぎる」だそうで……。泣き落としが効いたか(ちょっと違うような)？　堂領　輝昌(20)神奈川県

なんにせよ，ハードディスクが手に入ったのだからよかったですね。そんな奇特な友人を誰か私にも紹介していただけないでしょうか？

◆「SX-WINDOW ver.3.1」が届いた。徹夜で環境を整えた。2種に合格した。しかし……ハードディスクがほしい(ない)，CDプレーヤーがほしい(ない)，彼女がほしい(いない)，のでヨロシク。

西浦　宏和(20)愛知県

バイトする，お金を貯める，街に出て……。がんばってくださいね。

◆スキャナを買いました。夫婦で，互いの写真を取り込み，印刷し，それに「らくがお」をして楽しんでいます。　匹野　義博(30)大阪府

いったいどのような「らくがお」になるのでしょう。一度送ってみてくれませんか。

◆SXCON.Xが悪さをするのでFISH.Xがエラーを起こしてしまう。困ったものだ。でもインライン変換よりもお魚だよな，やっぱり。それが人の道というものだ，うんうん。

井村　英二(23)滋賀県

人の道とはいろんな道があるものですね。ただ，SXCON.Xを入れないと「SX-WINDOW ver.3.1」買った意味が薄いかも。

◆友人から電話があった。様子がおかしい。妙に明るいのだ。どうやらメモリを2Mバイトから10Mバイトに増設したらしい。うらやましいのでいろいろからかったが，ぜんぜん動じない。うーん，心まで広くなったのねん(笑)。

斉藤　修(26)宮城県

ここは，皆でメモリを12Mバイトにするし

◀岡村　直也　兵庫県
今月の「STUDIO X」にもありますが，女の子のユーザーって少ないですよね。でも，学が恵子のX68000を購入した理由を知ったら……。

◀吉田　淳一　宮城県
彼女のプロフィールを紹介しておきましょう。名前がPRO子，以前はのんきでしたが，頭の回転を速くしたらわがままになったそうです。

かないですね。マシンも安心，オマケに心の広さまでついてくるとは，これが正しい人の道(?)。ただし，借金に苦しんでも当方では関知いたしません。

◆6月上旬の2週間，母校に教育実習に行った。本当によい2週間だった。授業参観，研究授業，……と忙しかったが，文化祭など楽しいこともあった。しかし，なによりも忘れられないのは，1-7の生徒たち。最終日の放課後，教室に残ってくれて……。記念として，ネクタイピンをもらい，皆で歌をうたった。忘れられない2週間になりそうである。皆，がんばれよ(少し先生調)！
中村　学(22)福岡県

2カ月前の不安が嘘のようですね。皆さんでうたった歌がなんだったのか，ちょっと気になります。

◆福岡県に私と同姓同名の読者がいるらしい。その存在を電話帳以外で確認したのは初めてだが，同じOh!Xの読者であるのがちょっとうれしい。この分だとX68000百万台の野望が達成される頃には10人くらいはいるかもしれない。ということで「求む3人目！」もしいたら名乗り出てほしい(いないと思うけど)。同名さんなら4コマにいるけど……いっくぽーん。
中村　学(23)石川県

ということで，なんとなく並べてしまいました。

◆研修で東京に来ているのをいいことに，おもちゃショーに行ってきました。「SEGASATURN」とか見ていると，パソコンでゲームをする時代は終わったのだなあとつくづく思います。あのスペックで5万円を切るんですぜダンナ。ビジネスユースを除けば，パソコンは再びプログラムを組む機械に戻るんじゃなかろうかと思います。いい傾向ですね。それにしてもコモドールの倒産はショックでした。
越智　文昭(31)愛媛県

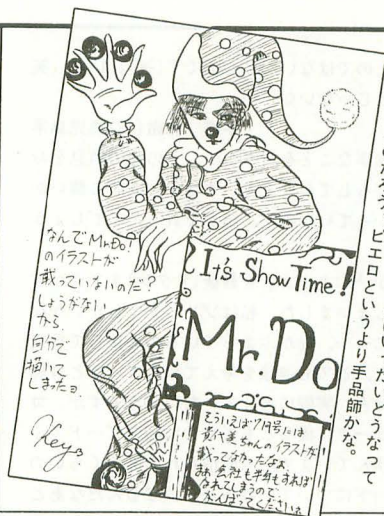
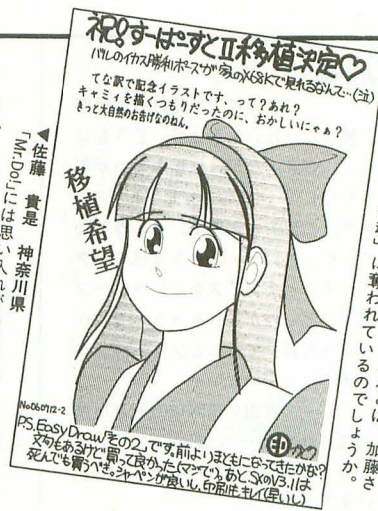
個人的には，プログラムを組むマシンをメーカーが出してくれるか，少し不安ですが……。

◆私もビジネスショウを見てきました。CASIOのブースのコンパニオンの衣装がよかったです(なにを見てきたんだか……)。初めてああいうショウを見に行ったのですが，スーツを着ていって正解でした。あんなところにTシャツ1枚じゃ不似合いですね。ちなみに，学校のオリエンテーションでうちの学科は全員行かされたんですけどね。
近藤　健一(18)神奈川県

しかし，学科の人が集団で移動していたら……怖い。

◆友人に愛車の原付を売って，MIDIボードと「Mu-1 GS」を買ってしまうことにしました。好きな音楽はできるし，徒歩通学で健康にもなって，まさに一石二鳥(?)。
風越　直紀(22)宮城県

あれ，音源は持っているんですよね(?)。

◆誕生日の夜に妹が「おめでとう！」といってシャーペンをくれました。その日に祝ってくれたのは妹だけでした。プレゼントよりもその気持ちが嬉しい。このハガキはもらったシャーペンで書いています。
小林　健一(21)群馬県

なんと健気な。でも，これで妹さんは海老で鯛を釣ることができたかもしれません。妹の誕生日には兄の威厳が問われますね。

◆7月号の発売日の前日，24歳を迎えた。昨年の前厄は急性アルコール中毒による入院に始まり，卒論再提出再発表に終わった。本厄の今年は……。さて，いい娘でもどこか落ちてないかな。
松嶋　竜(24)東京都

そんなよこしまな心を抱いていると……。

◆学校の自販機からミロが消えてしまいました。明治○○ッ○の自販機に入っていたのですが，ミロばかり売れて○○ッ○が売れないからのようです。ありょーん。
村上　洋樹(18)東京都

普通，売れないものが自動販売機からなくなるものだと思っていたのですが……。

◆眠いです。寝不足です。52試合全部見る気でいたけど，やっぱり無理みたいです。どこかの国では徹夜続きで死人が出たっていうし，アフリカでは戦争が休止してるっていうし，すごいですね。これが載る頃には優勝国の決まっているワールドカップのことでした。
三浦　貴至(22)埼玉県

それにしても，世界中でいろんなことがあったようです。たとえば，タイでは決勝戦の翌日の寝不足を心配して休校になったみたいです。日本の学生は眠い目をこすって，学校に行ったというのに……。

◆思わず，徹夜してしまった。ああ，鳥のさえずりが……。体がだるい。こんな体調で，はたしてテストを受けられるのか。あり，おり，はべり，いまそがり。趣がある。ぐぅ……。いとかなし。
森谷　好雄(17)北海道

このハガキが投函されているところをみると，テストは無事に受けられたようですね。はたして，その結果はいかに。

◆運動不足を感じ，風呂に入る前に筋トレをやり始めた。私は腹筋をすると，なぜかお尻の皮がむけてしまう。傷が治りかけで，とても痒くておサルモードに入っている。
喜多　清高(24)兵庫県

腹筋の鍛え方にもいくつかあるので，別の方法で試してみてはいかがでしょうか。まさか，剣山の上とかで腹筋をしてたりしませんよね。

◆このあいだの中間テストは，努力のかいあって，82点と久しぶりに優をとることができた。非常にうれしかったが，事件はそのあとに起きた。友達と本屋に行ったのだが，ちらっと見ると〝六三四の剣〞なる本があった。私は友達に「〝ろくさんし〞ってなに？」とボケた質問をしてしまった。友達に〝六三四〞を〝むさし〞ということを聞き，自分の国語力の乏しさにショックをうけた。偏微分のテイラー展開ができるより〝むさし〞と読めるほうがいいよなぁ。
室谷　由久(17)富山県

〝六三四〞が読めなくても大丈夫だと思います。〝武蔵〞が読めないと困るかもしれないですけど。それよりも，偏微分のテイラー展開ができるほうが，いいと思いますよ。

◆付録ディスクが白いページに貼りつけてあり，少々驚きました。そのページにはミシン目があり切り取れるはずが，その上にディスクの入った袋が貼ってあり，結局そのページは破ってしまいました。以前みたいに，はさみ込んであるほうがよかったような気がします。それと，このハガキ，印刷がずれてるような……(なぜか不幸なOh!Xを買ってしまったようです)。
下倉　雅行(20)岐阜県

なぜ，ディスクが貼ってあったかは，次のハガキをお読みください。

◆6月より，雑誌に3.5インチディスクがつけられるようになった代わりに，雑誌内にくっついていなければならなくなったようですね。ページの途中にディスクがついている雑誌もありましたが，Oh!Xの方式はなかなかスマートでよかったと思います。
菰田　英和(24)愛知県

10月号ではまた少し変わりそうです。今度も気に入っていただけるといいのですが。ということはディスクが……(予告参照)。

◆私の7月号は最終ページにオマケがついていましたが，皆さんのにはオマケがついていましたか？
田中　純志(21)愛知県

ハズレくじなしのプレゼント企画でした(大嘘)。

▲佐藤　貴是　神奈川県
「Mr.Do!」には思い入れがあったみたいですね。しかし，帽子の先のボンボンはいったいどうなっているのだろう。ピエロというより手品師かな。

▲加藤　信夫　宮城県
「Easydraw」を使った第2弾です。「スパII」の移植決定記念イラストなのにナコルルとは。加藤さんの心は「侍魂」に奪われているのでしょうか。

◆なんか天井が低いなあ。

大森　亮寛(18)愛媛県

皆さん，狭苦しい思いをさせて，どうもごめんなさい。でも，なかには無事なハガキもありましたし……(見苦しい)。

◆付録ディスク，ご苦労さまです。ところでいつも思うのですが，付録ディスクを厚紙のワクからはずそうとすると，ディスクエンベロープの「品質保証」のところが破れちゃうんですが……，ハッ，富士写真フイルムとソフトバンクの陰謀か！　そうにちがいないっ！

有山　茂芳(20)神奈川県

そんなに疑われても……。素直にカッターなんかで切り取っていただけるとよろしいかと思います。

◆家の周りにいくつかあった水田も，いまではほとんどなくなりました。それでも夜になるとカエルの声が聞こえてきます。その合唱は，まるで指揮者がいるかのように，一斉に始まり一斉に止みます。カエルにも「群れ」的意識があるのでしょうか。　中島　民哉(23)埼玉県

陰に隠れてこっそり見てみると，実はおたまじゃくしが楽譜を作って……。

◆ある朝，新聞を取りに外へ出たら，戸袋につばめが5羽とまっていました。親が2羽，ひなが3羽，ずんぐりして大きいのがひなのようでした。やっと巣立つのが嬉しくて，パジャマのまま飛び去るまでにらめっこさせてもらいました。　下田　達也(27)三重県

にらめっこで笑ってしまい，落ちて死んだなんてことにならずによかったですね。

◆よくフランス料理に「舌平目」なる魚が出てきますが，私の地元ではそれを「ゲタ」と呼んでいます。県外から来たある人が，その名前を忘れ，魚屋で「ゾウリ」くださいといって大恥をかいたという話が新聞に載っていたのですが，私もどちらかというと「ゾウリ」のほうがあってるなぁと思いながら笑ってしまいました。

藤原　彰人(24)岡山県

これは聞いたことはなかったのですが，そういえば鮫のことをワニとかいいますよね。

◆7月号のアヒルですが，最近みかけません。農学部のどこかのサークルの鍋の具になってしまったのではないかと心配です(笑)。いや，笑いごとじゃないな……。

手嶋　和徹(22)鹿児島県

無事なことを祈りつつ，その後の消息をお待ちしております。やはり，人間に襲いかかっていたのが，怒りを買ったのでしょうか。

◆妻が「ぷよぷよ」を最後(サタン)までクリアしてしまいました。私はゾンビまでしかいかないのに……。妻がぷよっているのを後ろで見ていると，とても連鎖を考えて積んでいるとは思えません。実際に考えていないそうですが。コンピュータに優るとも劣らない超スピードでぷよを積んでいます。人間慣れればあのくらいのスピードについていけるようになるんだなあと思いました。　後迫　浩一(33)神奈川県

すでに人間ではなくなって……奥さんの背中に羽が生えたりしていませんか。

◆$\sqrt{N}$(ルートN)の法則。航空材など各社共同開発の場合の開発費は……3社共同の場合→総合計開発費は $\sqrt{3}=1.732$倍……となる。つまり，N社の総合計開発費は約$\sqrt{N}$倍となってしまうが，N社で割ると1社当たりの開発費は結局安くなるということだそうです。

迫田　賢一(40)大阪府

感覚的にはわかるものの，そんな法則があるとは……おもしろいものですね。

◆(ふ)さん，女性が2％で少ないなんて甘い。鉄道ファンに占める女性の割合は0.3%だ！　鉄チャン300人をかき集めても，そこに女性は1人いるかいないか。Oh!Xとどっちがマシ？　さあさあさあ。挑発的な言い方になってえらいすんまへん。でも，きっとどんな男女比も鉄道よりはマシ……。　新宮　智子(27)神奈川県

先日の結果はOh!PCのものです。Oh!Xでは，2％を割り込んでいます。でも，0.3%とは，かなり少ないですが，もっと悲惨な状況のものってあるんでしょうか？

◆引っ越すときのもの減らしって，どうすればいいのでしょうか。ほとんどがないと困るし，実際使ってるんですよね。ともあれ，自分の住むところは慎重に選びたいものです。ところで，岡村さんの4コマは新シリーズも実に面白いです。7月号のものは，会社で読んで大笑いしてしまいました。　関本　正人(22)長崎県

使っているのなら，減らす必要はないのではないかと思います。あとは，もっと広い部屋を探します？

◆大学内の駐輪場で鍵をかけておいた自分の自転車が盗まれた。大阪の風土が悪いのか，うちの大学生が人としてのデキが悪いのか，もしくは鍵を1個しかつけなかった自分が悪いのか。とりあえず，半年後同じ大学に通っている自信がないので，下宿先での定期購読はできそうにありません。　山本　哲也(19)大阪府

ある人の話では，東京の東端で盗まれた自転車が数カ月後に見つかり，警察から問い合わせの電話があったそうです。なんと東京の最西端。たくさんの人に乗り継がれてたどり着いたようです。東も西も似たようなもの。なんか悲しいですね。

◆6月20日，Oh!Xが届いた。「今月は付録ディスクはなしか……」。真ん中あたりで折れ曲がった包みを見ながら思った。「なんか分厚いな……お，おい，これは……」。折れ曲がった包みの中から見覚えのある薄い包み。出てきたのは，「クシャ'フロッピーディスク'」だった。すぐにオペに取りかかった。軟盤移植手術……ディスクの外側を切り取り，中の円盤部分だけを取り出し，ほかの折れていないフロッピーディスクに入れてディスクコピーをとった。めでたし，めでたし。　川波　弘(29)愛媛県

ハガキにはディスクの絵が描かれていて，楽しく読ませていただきました。ディスクの中の円盤って結構耐久性がありますね。

◆「ない，ない，なーい！」。大学の落とし物コーナーの前で動揺を隠しきれなかった僕は，心の中でそう叫んだ。さっそく，落とし物担当の課へ問い合わせた僕に，事務のお姉さんがそっけなく答えた。「あぁ，あれならどうしてもほしいという人がいたので，期限が切れるのを待って，その人に譲りましたけど」ガーン。欲しがる奴までいたなんて……。すっかり期待を裏切られた僕は，このことはやっぱり彼女(M子さん)にいうまいと誓うのでした。

大畑　佳史(21)兵庫県

そこまでこだわるところをみると，案外自分がほしかったのでは……。

◆7月号の発売日，東京ドームでの巨人戦の試合開始前にライトスタンドでOh!Xを読んでいた方へ。あのとき目の前に立っていた係員は私です。今度見かけたら声をかけてください(といっても覚えているわけないか)。

杉山　洋之(21)東京都

仕事の関係でバイク便を利用しています。かなり前の話ですが，その配達のお兄さんに，「Oh!Xの読者なんです」といきなりいわれて，ちょっとびっくり。読者の方って，いろんなところにいるんですね。外で読むときは気をつけないと……。

◆今年で30だぞー。子供が2人，パソコンが10台，ポケコンが4台，バイクが1台。さて，こ

◀鈴木　道明　埼玉県
「奇々怪界」とタイトルは不気味だけど，キャラクターが可愛いですよね。でも，このイラストは，ゲームのキャラクター以上に可愛いですね。

◀溝畑　知幸　兵庫県
しばらく日本を離れるそうですが，イラストもしばらくお休みになるのでしょうか。ちょっと残念ですが，体に気をつけてがんばってくださいね。

れからなにが増えるだろうか？
北川　浩樹(29)岐阜県
……借金？

◆子供が産まれました。ただいま４カ月を過ぎ，手にするものはなんでも口に入れたり，なめたりの毎日。それに歯がはえてくるのか，カミカミもします。主人はちらかし魔，フロッピー，マウス，本(もちろんOh!X)，なんでもそのへんにポイポイ。近い将来，これらのものがよだれの洗礼を受けるのは必至かな。
森本　幸子(27)広島県
まずは，ご主人を教育し直しますか。

◆あのスリッパ(７月号参照)は妹にあげました。(だって，私の理想の女性(４月号参照)には足がないもの)。さて，最近恐ろしいことがありました。某大型店で４歳児(推定)が「スパII」でザンギを使って次々に勝ち進んでいたのです(対COM戦)。「ついにここまできたか，４歳児(推定)」と思っていたら，単にその子供は対戦台の2P側でコンパネをいじくってただけなのです(笑)。これだからクソガキは面白い。
平野　鉄之助(18)長野県
1P側に座っていたのは，実はその子供の友達だったりして……。

◆茨城の水戸市に引っ越して約２カ月。コンピュータ店がみつからない。誰かいいところを教えてくれませんか？　安海　高明(20)茨城県
休みを利用して近くを探し回ってみるか，地元のユーザーを探してみてますか。それともあきらめて，都心に出てきます？

◆近くにあった某PCショップがつぶれ，跡地に新しい店ができた。またもPCショップ。もしかして，のっとられたんじゃないの？　店の人々は人が良さそうなんだけど……なぜか怖い。金魚のフンのようについてくるし，「離れてくれ」って思ってしまう。この店どう思います？
石井　義尚(18)神奈川県
仮面を被った宇宙人が……。

◆１年前から親元にX68000PROを置き去りにしていたが，ようやく２台目を購入した。現在は去年の５月号から，もう一度読み返しているところです。　弥益　慎司(21)東京都
親元に置き去りにされたX68000PROは無事なのでしょうか？　ホコリにまみれたり，押し入れの中で泣いていたり……していませんよね。

◆突然ディスクつきで850円，サイフの中は……。７月まで生きてゆけるのだろうか？　予告は毎月見ておくのだと痛感する今日この頃(6/18某書店にて)。　千葉　茂夫(22)埼玉県
９月号の値上がりは予告ではわかりませんでした。今後は編集後記(SHIFT・BREAK)も読むことをお勧めします。

◆うーむ，「ジオグラフシール」のシークレットボーナスって意外とわかってないんだね。私はすぐに見つけられましたが……。変かな？　ちなみに，ジャンプなしクリアで50万点，ショットなしクリアで30万点です(１回でもジャンプ，ショットしちゃだめ)。後者はかなりムズかしいと思うけど，点数はたいしたことないんですよね。前者は１面で楽にできます。さあ，ほかのも探そっと。　片倉　純也(19)宮城県
ダンジョンの面をクリアすると知らないうちにボーナスが入っていたのは，そういう訳だったんですね。

▲平　智征　神奈川県
じっと前を見つめる彼女の目が強い意思を感じさせます。羽のふわっとした感じもとってもよくて，すっかり気に入ってしまいました。

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

仲間

★「Lovers」では，新規会員を募集します。入会資格はＸ１/turboシリーズの５インチ２Dか２HDドライブを２台お持ちの方であれば結構です。２カ月に１回ディスクマガジンを発行しています。内容は会員のフリートークや音楽，プログラム，CGなどです。会報のサンプルと案内書をご希望の方は500円分の定額小為替を，案内書のみをご希望の方は80円切手を同封のうえ，以下の住所まで送ってください。〒302-01　茨城県北相馬郡守谷町守谷甲2779-109　高橋　顕治(27)

★「MANATEE」という名前のX68000用ディスクマガジンを趣味で編集しています。方向性や内容について取り決めはありません。参加の意思をおもちの方はフォーマット済みの２HDディスク４枚と，切手を貼った返信用封筒を同封のうえ，以下の住所までお送りください。3.5インチも取り扱っています。〒305　茨城県つくば市春日4-13-30 明峰ハイツA-111　野口　友則(22)

★X68000用のディスクマガジン「Mecca Tick」を発行するにあたり，会員を募集します。内容はMIDIデータやプログラムを満載する予定です。一度見てみたいと思われる方は，フォーマット済みの５インチ２HD１枚と宛先を書いた返送用封筒に130円切手を貼って，下記までお送りください。要２Mバイトです。3.5インチには対応していません。〒577　大阪府東大阪市下小阪1-15-13 ドミトリー小阪203　小早川　大吾

売ります

★カラーイメージユニット「CZ-6VT1」(グレー)と「TS-VTBOX」をセットで35,000円くらい，キャノンのプリンタ「BJC-820C」とカラーインクをセットで45,000円くらい，ビデオプリンタ「CZ-6PV1」を30,000円くらいで売ります(すべて送料込み)。連絡は往復ハガキでお願いします。〒308　茨城県下館市小川1385-7　鯨　雅之(36)

★カラーイメージスキャナ「CZ-8NS1」とアイ・オー・データ機器のスキャナ用パラレルボード「SH-6BN1」を30,000円くらいで売ります。箱はなしで，説明書，付属品はあります。連絡は希望価格を書いて往復ハガキでお願いします。〒198　東京都青梅市河辺町1-906-4 日神パレステージ河辺第２-103　佐々木　博之(36)

★MIDIボード「CZ-6BM1」を5,000円，２Mバイト増設RAMボード「CZ-6BE2」を10,000円，ロジテックの100Mバイトハードディスク「LHD-FM100E」を15,000円で売ります。連絡は往復ハガキでお願いします。〒791　愛媛県松山市東山町4086-11　蔵本　健一(26)

★RGBシステムチューナー「CZ-6TU」を15,000円(送料込み)で売ります。テレビコントロールケーブルつき。ただし，箱はありません。連絡は往復ハガキでお願いします。〒615　京都府京都市西京区桂艮町25-29　市田　治男(66)

買います

★カラーイメージユニット「CZ-6VT1」を45,000円で買います。付属品があれば，説明書と箱はなくてもかまいません。連絡は官製ハガキでお願いします。〒350-13　埼玉県狭山市上奥富1113-10　坂東　宏二(21)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今回から，新しく第10期愛読者モニタの皆さんが登場します。今月は7月号の内容に関するレポートです。

●ここ数年，GS音源を始めとするシンセサイザは激的に安くなり，「万人のDTM」もかなり近いものになったと思います。しかし，まだ，安くなったからといってすべての人が手にできる範囲でないのは事実です。特に興味をもっていない人にとっては，高い出費でしょう。7月号の特集「入門コンピュータミュージック」は，ひとりでも多くの人に興味をもってもらうという意味では，最もよい形の特集であったと思います。

特集の中ではZPLKがもっとも興味を引く記事でした。AD PCMをPCMに変換することで，AD PCMの可能性が広がったといえるのでは？ と思い，なにか作ってみようとしましたけど，なかなかアイデアが出ませんね。

奥田 直也(21) X68030/X68000 SUPER/ACE-HD/MSX2/PC-E550 愛知県

●現在のコンピュータミュージックは，グラフィックなどと比べて，技術的，知識的に敷居が高いと思います。せっかくコンピュータがあるのですから，せめて入門だけでも，もう少し平易にならないものでしょうか。細かなテクニックは無視してメロディ，リズム，コードを簡単に組み立てられるようなツールがあれば，と思うのですが。もっと落書き感覚で遊べるツールが欲しいですね。

弦元 達也(23) X68000 ACE-HD 香川県

●私個人としては，7月号の特集は非常に興味のある分野でよかったと思います。しかし「入門コンピュータミュージック」というタイトルの割に，内容的にはやや難しい感じを受けました。Z-MUSICシステムを例にとっても，Z-MUSICをシステムに組み込むことができ，データファイルが存在してても，どのようなコマンドを与えれば音が鳴るかわからないという人(実は私のことです。最近Oh!Xのバックナンバーをひもといて問題は解決しました）もいるのではないでしょうか。

あと，基本的なことは理解しているつもりだったPCM。しかし，本当にわかっていたのは概念だけだということを，改めて思い知らされた記事「基本はPCM，あとはみんなついてくる」が一番印象に残っています。AD変換の基本的なことは知っているつもりでしたが，この記事を読んで，いろいろと新しいことを発見しました。ΔPCM，DPCM，AD PCMなどは，理論的な解説ではありますが，非常に興味深く読むことができました。

また，音量の変化，音程の変化など，AD PCMをからめたところはちょっと難しかったのですが，それぞれにサンプルプログラムが紹介されていて，とても参考になりました。

壁谷 喜嗣(35) X68000 EXPERT/PC-9821 As/9801NS/E 愛知県

●特別付録「CGA入門キットGENIE」は素晴らしいですね。非常に簡単にメカが作れます。これならいままでCGAシステムを難しいといった理由で敬遠していた人も，楽にCGAの世界に浸れるかもしれません。本文の解説もわかりやすく，これなら初心者も安心です。CGAの初心者である私がいうのですから間違いありません（ただ，上級者の人たちには，ちょっともの足りなかったかもしれませんね)。連載のほうは来月号に続くようなのでいまから楽しみにしています。7月号の付録ディスクで遊んでいるぶんにはなんの不満もありませんでした。

大上 幸宏(21) X68000 PROII 鹿児島県

●江口響子の実用講座「夏のカードをフォトコラージュで」は，「X68000でもCD-ROMが扱えるんだぞ」といういい見本になったと思います。確かに本体内蔵でないのは，ある意味うっとおしいですし，CD-ROMソフトも出にくいはずです。

しかし，こういったサポートはシャープじきじきに行ってほしいものです。OSのSCSI制御（SCSIDRV.SYSなど）で，CD-ROMのサポートぐらいはしてもいいんじゃないでしょうか。そうそう，次回（もしもあれば）はやはり「イメージスキャナで……」といった感じの講座はどうでしょうか。MATIERでSCSI転送もサポートしていることですし。ビデオ入力ユニットもいいですね。やっぱりX68000って自家製のものを作れるマシンだなあ。

中矢 史朗(23) X68030/X68000 ACE/PC-386 愛媛県

●「"実戦!"ゲーム作りのKNOW HOW～基本セオリー編」で扱っていた，スティック判定と方向検索などに使用するDDAアルゴリズムの解説は，非常に興味深く読ませていただきました。「こういう記事を待っていたんだよ」ってところですか。自分の周りに自分より上級のプログラマがいない人にとって，一番不足しがちなのがこういう小手先のテクニックや基本アルゴリズムでしょう。よく「テクニックは人から盗むものだ」なんていいますが，盗みようがない場合もあります（デバッガやディスアセンブラなんてものもあるけど，それだけで勉強しろっていうのも，結構酷な話だと思いません？）。できれば，ゲームにかぎらず，こういう小さいテクニックをチビチビと解説する連載を始めてほしいですね。特に執筆者にこだわらず，1ページでも2ページでもかまいませんから。

渡辺 祐介(19) X68000 富山県

●最低でも楽譜を前提とすること。オヤジの鼻歌くらいに気軽に始められること。理想のDTM環境といえばこれぐらいは最低条件でしょう。現在のDTMは「素晴らしい音楽家」方向のベクトルしかもっていない（ような気がする)。スーパーリアリズム以外許されない絵画のようなものかな。まあ，私は生楽器や肉声をコンピュータで実現してほしいとは思わないけど。それに，MMLのような中間言語というかRISCチップのマシン語コードのようなものは，はやいところ単なる内部表現になってほしいものです。

あと，最近「THE SENTINEL」がひそかにいい連載を連発してくれているので，なんか嬉しい。マシン語講座もシューティングゲーム作成講座もきちんと入門記事していて「基本的なことはわかっているとする」などといって，いちばんやっかいなところを放り出したりしていないのがいいですね。

石田 伯仁(21) X68030/PC-9801VMII/PC-8801mkIIMR 神奈川県

ごめんなさいのコーナー

7月号 The World of X68000II

P.33 ゲームレビュー中，「C力検査」の制作者の名前が間違っていました。正しくは小林康弘氏です。小林氏ならびに関係者各位に，ご迷惑をおかけしましたことをおわびします。

バグに関するお問い合わせは

☎03(5642)8182(直通)

月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

進化する早川式繰り出し鉛筆

▶大幅なバージョンアップとまではいきませんが，着実に進化し続けるSX-WINDOW。今度はver.3.1となって，いままで好評だったシャーペンが，インライン変換，ドローデータのサポートなどの機能強化され，さらに使いやすくなっているのがいいですね。

各種アプリケーションもそれなりに揃いつつあるし，8月号で瀧氏の改造を行えば，特別なハードなしに1024×1024ドット表示までできてしまう。コンソールウィンドウも標準サポートされたので，プログラミング環境もずいぶん整いました。

それにしても，ウィンドウシステムは資源を消費しますね。CPUパワー，メモリ，外部記憶装置などなど……ありすぎる分には困りません。困るのは，ユーザーの懐のみ。でも，便利に使いたいなら贅沢にならなきゃね。学生諸君！　夏休みのバイト代を，会社員の方はボーナスの残りを注ぎ込んでハードウェア環境整備に，そして頭を使ってソフトウェア環境作りに精を出してみませんか。

▶1994年6月号で紹介した「POLYPHON-24」を販売しているネオコンピュータシステムが，現在連絡の取れない状態になっております。編集部では，この件に関して調査をしていますので，商品が届かないなどの苦情のお問い合わせをいただいても，まだなにもお答えできない状況です。新たな連絡先が判明したときには，誌面でお知らせしたいと思います。読者の方々にご迷惑をかけたことをおわびいたします。

▶「X68000マシン語プログラミング」は，“著者締メ切り間ニ合ワズ”のため，「“実戦！”ゲーム作りのKNOW HOW」は，著者多忙につきお休みとなってしまいました。楽しみにしていただいている読者の皆さんには，本当にもうしわけありませんでした。

▶最後に来月号の付録ディスクに収録予定のものは……あ，スペースがない。ということで，予告を見ながら楽しみに待っていてください。

投稿応募要領

- ●原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- ●プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ●ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。
- ●投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒103 東京都中央区日本橋浜町3-42-3
ソフトバンク出版部
Oh!X「テーマ名」係

SHIFT・BREAK

▶「ぶうう〜ん」，扇風機に当たりながら寝ると本当に死んでしまうのだろうか？　このまま寝て明日の朝，再び目覚めることなく冷凍イカのように固まってしまうのではないか，という不安感は近頃の退屈気味な生活に変化を与え得るもので，私は胸が踊る思いだ。しかし，私はおフランス製のベッドの上で寿命を迎えるという密かな夢を抱いている。（H）

▶先月明けの宴会でわかったことがひとつ。この編集部での一番の大喰らいは，某（ふ）さんであることがわかった。判定方法はいたって簡単，各自食べた量を自分の体重で割ればいいだけ。私は体重88kgで（ふ）さんの2倍。食べる量はどう考えても1.5倍ぐらい。体が大きいといっても，さすがに2倍は食べられないもんね。（瀧）

▶ぼくは，しゃべることばがむずかしいと，よくいわれます。しんぶんに出てくるような単語が，わりと多いです。このまえ，「お前，難しい単語使うね」といわれたので，「いやー，なるべく平易な言葉で喋るようにしてるんですけど」と答えたら，みんなに笑われました。そういうことばがしぜんに頭に浮かんでくるので，直すのはたいへんです。（E.K）

▶最近，「Vampire」とゆーゲームでよく遊ぶ。しかしこのゲーム，やっているとガロンを見ては「らぶりーわんわん（狼だっ！）」とか，フェリシアを見ては「あ，猫の後始末のポーズ」とかよくわからんネタばかりが思いつく。このゲーム，よっぽどおいらのお笑い線をくすぐるのか，それともおいらがそーゆー体質になってしまったのか？（で）

▶とうとうDOS/Vマシンを購入。ゲーム専用と思って買ったが，WINDOWSには使いものになるフォントが標準で揃っていたため，図版作成はあっさりWINDOWSベースに移行した。ウィンドウシステムとしての使い勝手はSX-WINDOWがずっと上だと思うが，結局はこういうつまらない部分で差がついているというのは納得いかないものだ。（A.T.）

▶海外へ行くと，日本人旅行者はすぐわかる。後ろから見ればわかる。歩き方に締まりがないのだ。若い人ほどだらだらと歩いている。同行者は「平和ボケだね」といっていた。英国人もすぐわかる。目が合ったとき，わざとらしい笑顔を見せたらそうだ。あの妙な親切さは旅行者にはうれしい。インド系女性はみな美人。東洋系の旅行者が一番無愛想。（K）

▶自宅療養期間がとけて出勤を許されるようになった。ただし、体調は日増しに悪くなっていくようだ。これを主治医に話したら、薬の量を増やされた。うーん、薬漬けの毎日だ。完治までにはまだ遠いということを実感する。Sはつまらないと先月書いたが1クール終了あたりから盛り上がってきたようだ。原作がある程度進むのを待っていたのかな。（KO）

▶先日，我が家にもレーザーディスクが入った。しかし，パナソニックの製品ということだけで型版さえも覚えていない。動かしたのも一度だけ。なぜかといえば，自分で買ったものではないからだ。なんと会社のパーティ（？）で当たったのだ。ラッキー。でも，忙しくて全然見ている暇がない。早く仕事を終わらせて，ソフトでも買いに行こうかな。（高）

▶高校野球には興味がないのだが，何気なく眺めていた新聞で，ふと，18歳まで住んでいた都市に知らない高校があるのに気がついた。いつできたのかは，家族の話題にもなっていなかったのでわからない。このあいだ隣接の町だか村だかを合併したそうだし，人口も増えたんだろうな。そうか，ひとはこうして故郷を偲ぶものなのか，とちょっと思った。（ふ）

▶企画進行中のゲーム制作のために，アセンブラでゲームを作ったことのあるプログラマを募集します（SION4とは別モノ）。東京近郊に在住，週に1回は編集部まで来れる人で，比較的暇な学生がいいですね（スケジュール的にきついので）。ウデに自信がある人は，編集部の山田まで電話でご連絡ください（PM5:00〜8:00に受けつけています）。（J）

▶VHSのビデオデッキが故障して数カ月。全然困っていないのだが，2度目の修理ともなると修理に出すのも億劫で，新しいデッキを買いたくなってきている（まあ，修理には出すのだろうが）。時間を都合して，お金を用意して，機種と買う店を決めて，「さあ，行くぞ」という段になるといまいち決心が鈍る。置き場所を確保するのが先か？（U）

▶肩を強打して痛みが引かない。水曜日の朝イチで病院へ行くと，今月から午前の診療はなくなったといわれる。翌日の午後に行くと今度は整形外科の診療は火，水，金のみだという。そして金曜日。2時間待っても自分の番が回ってこない。しかし文句ひとついわずに待っているお爺さん，お婆さんたちって元気だなあ。チケット取るより大変だぞ。（T）

microOdyssey

3DOマシンを使ってみて数カ月。基本的な性格についてはだいたい思ったとおりだったが，世間の評価は，だいたい「遅くて，絵が汚い」というところで一致するだろう。高速CPU＋グラフィック強化ということで登場したマシンのはずなのに，(処理内容はともかく)快適な速度で動くソフトというのが非常に少ない。

速度は，例によってプログラマのスキルに依存するところが大きいようだ。3D処理といっても，見るからに重いものから軽々とこなすものまでソフトによってさまざまだし，CD-ROMの待ち時間もずいぶん差が見られる。

絵が汚いのは動画方式に問題がある。現在はCINEPAQという方式で圧縮されたものが使用されている。これはMacintoshやWindowsで主流になっているものだが，CD-ROMからの転送となると輪郭ガタガタ，画面はノイズだらけ，階調はとてもフルカラーのフレームバッファに出しているとは思えないくらいガタ落ちになる。フレーム数も秒間10フレーム程度しか出ていないように思われる(ものによるが)。

MPEG1が導入されればある程度解決される問題ではあるが，気をつけたいのはMPEGというのはデータの作り方で画質が露骨に変わってくるということだ。

投入されるデータ量から考えれば，とてもまともな画像が出力できるとは思えないものだが，エレショウなどの出展映像ではMPEG1の画像でもほとんど粗は見えない。しかし，市場に出回っているデータはかなりひどい。X68000用のMPEGボードのデモを見た人なら，概ね制作年月日の古いデータほど粗い画調になっているのがわかると思う。これは，年々，新しいアルゴリズムによるエンコードが行われているからである。最良の映像になると，場面ごとに最適なアルゴリズムでエンコードされている。

いくつかの次世代ゲーム機ではJPEGでの動画が行われるようだ。雑誌によってはMPEGに対し「JPEGは綺麗で高圧縮」と持ち上げているが，どう考えてもアルゴリズム的に時系列情報を利用したMPEGにかなうはずはない。

CD-ROMのデータ転送速度は秒間300Kバイト程度だ。すなわち1フレームあたり10Kバイトのデータとなる。JPEGで10Kバイトにまで縮めた絵の画質なら簡単に想像がつくだろう。

一方，最近，VOD(Video On Demand)という言葉を目にすることが増えてきた。見たいビデオがすぐ見れるというサービスのことだが，アメリカではすでにケーブルテレビ回線を使ったVODが実用化されている。蓋を開けてみれば，レンタルビデオ屋のにーちゃんが注文が入るたびに，ビデオを回して各家庭に送っているにすぎないが。

これをデジタル回線ですべてデジタルデータで行うというのがあるべきVODの姿である。1週間分の全テレビ放送が蓄積されたり，映画のライブラリが見たいときに手にできる。

データはすべてハードディスクで蓄える。MPEG1でも1時間あたり1Gバイト程度。クオリティ考えれば最低でもMPEG2になるが，これだと10Gバイトくらいになるだろうか。映画1本で20Gバイト……。その前に各家庭にデジタル回線が必要となる。LDボックスを買わなくてよくなる日は遠いかもしれない。　(U)

1994年10月号 9月17日(土)発売

特別企画　紅葉狩りPRO-68K

・使えるSX-WINDOWデスクトップアクセサリ
・最新版SX-BASIC　・X-BASIC用外部関数
・その他，小物ツールをふんだんに収録

新製品紹介　H.A.R.P

スーパーストリートファイターII速攻レビュー

特別付録　5"2HD　予価 950円

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(3233)3312
	〃	書泉ブックマートB1	03(3294)0011
	〃	書泉グランデ5F	03(3295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(3257)2660
	八重洲	八重洲ブックセンター3F	03(3281)1811
	新宿	紀伊国屋書店本店	03(3354)0131
	高田馬場	未来堂書店	03(3209)0656
	渋谷	大盛堂書店	03(3463)0511
	池袋	旭屋書店池袋店	03(3986)0311
	八王子	くまざわ書店八王子本店	0426(25)1201
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	043(224)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコンΣ上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある『新規』『継続』のいずれかに○をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください。なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

基本的に，定期購読に関することは販売局で一括して行っています。住所変更など問題が生じた場合は，Oh!X編集部ではなくソフトバンク販売局へお問い合わせください。

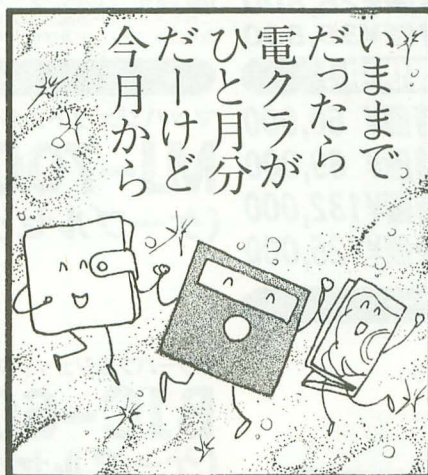
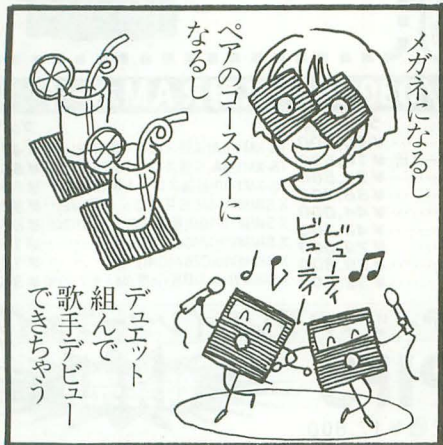
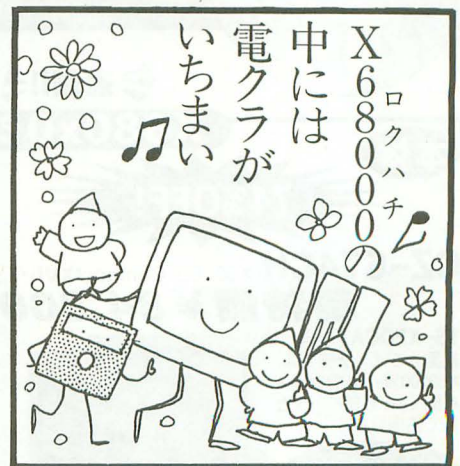
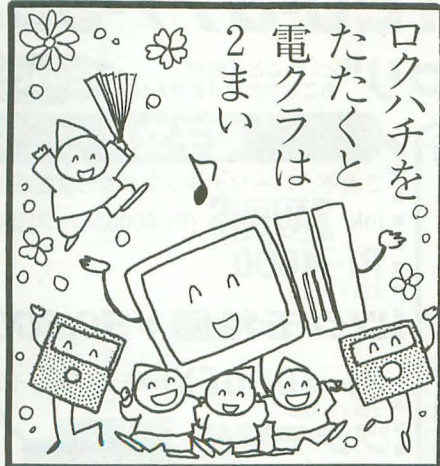
海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700

Oh!X 9月号
■1994年9月1日発行　定価680円(本体660円)
■発行人　橋本五郎
■編集人　稲葉俊夫
■発売元　ソフトバンク株式会社
■出版事業部　〒103 東京都中央区日本橋浜町3-42-3
Oh!X編集部 ☎03(5642)8122
販売局 ☎03(5642)8100 FAX 03(5641)3424
広告局 ☎03(5642)8111
■印　刷　凸版印刷株式会社

満開の電子ちゃん

作・え　岡村祭

75号(7/18発売)は、「ザウルスとX68をつなぐ!!」特集など盛りだくさんの企画です。

講読方法：定期購読もしくはソフトベンダーTAKERUでお買い求めいただけます。

★定期購読の場合＝購読料第75号（94年6月号）より6ヶ月分8,500円（送料サービス、消費税込）を、現金書留または郵便振替で下記の宛先へお送り下さい。

現金書留の場合：〒171　東京都豊島区長崎1-28-23 Muse西池袋2F　㈱満開製作所

郵便振替の場合：東京　5－362847　㈱満開製作所

- ご注文の際は、郵便番号・住所・氏名・電話番号を忘れずに記入して下さい。
- 3.5インチディスク版をご希望の方は、「3.5インチ版」とご指定下さい。
- 新規購読の方は「新規」と明記して下さい。なお、特に購読開始号のご指定がない場合は既刊の最新号からお送りいたします。
- 製品の性格上返品には応じられませんが、お申し出があれば定期購読を解約し残金をお返しします。

★TAKERUでお求めの場合＝1部につき1,600円（消費税込）です。

- 定期購読版と内容が一部異なる場合があります。御了承下さい。
- お問い合わせ先　TEL(03)3554－9282（月～金　午前11時～午後6時）

（なお、定期購読版のバックナンバーについては定期購読の方のみご注文を承ります）

幸俊威（大阪府）

今月も18日がやって来た。電源オンですぐ起動、マウスひとつでらくらく操作でおなじみの電脳倶楽部のアヤシイ水色の封筒が送られてくる日である。

送られてきたディスクは、僕の黄金の指によってドライブの中に押し込まれ、何十回、何百回、何千回…と、数えきれぬほどアクセスさせられ、悲鳴をあげる。しかし、これは電脳倶楽部として生まれたディスクの名誉の負傷なのだ。

右の写真は、まだ高校生の頃の僕だ。今はもう少し痩せている。

購売を始めてからというもの、オモシロイ奴になったと評判だ。

ラジオCM、好評 ON AIR中!!
ニッポン放送「伊集院光のoh!デカナイト」内
（木） PM10:30ごろ
海ラジオ 「大内ひろのしんの今夜もやっぱりFUNKYパジャマ」内
（金） PM11:00ごろ
FMノースウェーブ「SUPER CHOICE」
（金） PM7:30ごろ
ージガール越智静香さんが茶目っ気たっぷりにパソコンをおねだり。
は特価品をズバリ！ツクモ特価でお知らせします!!
聴いたら、ツクモに来ないではいられません～

12回払い、7.5%がナント6%に!

クレジット金利がこんなにお安くなりました! ～月々ムリのないお支払い額で欲しかったパソコンがお手元に!!～

支払回数(回)	1	3	6	10	12	15	18	20	24	30	36	42	48	54	60
分割払い手数料率(%)	2.5	3.5	4.5	5.5	6	9	11.0	12	12.5	16.5	17.5	22	23	28.5	29.5

TSUKUMO TSUKUMO TSUKUMO TSUKUMO TSUKUMO TSUKU

受付時間(平日)AM10:45～PM7:30
(日・祝) AM10:15～PM7:00
木曜定休

『FAX24時間お見積もり受付』
03-3255-4199
お名前.住所.電話番号.FAX番号をご記入の上ご依頼下さい。

ツクモグローバルJCBカード

JCBならではの国内・海外サービスにツクモオリジナルの特典をプラス。ツクモ各店にある入会申込書にてお申し込み下さい。くわしくはグローバル事務局03(3251)9898又は各店へ。
※ジャックス・VISA・セントラル・マスターも取り扱っております。

動画を始めてみませんか？

ビデオ入力ユニット **CZ-6VS1** 定価￥178,000

MC68EC020(25MHz)の32BitMPUを搭載し、SCSIインターフェイスを介してパソコンへデータを転送。動画・静止画を簡単に保存出来るアプリケーションソフト「ライブスキャン」を標準装備。1,677万色まで対応し、最大640×480ドットの高解像度で、高速取り込が可能です。但し680x0シリーズでご使用の場合には6万5千色までの表示となります。

ツクモ特価￥142,000

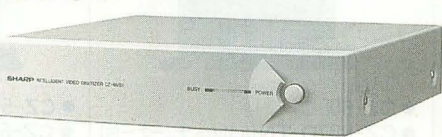
コンピュータアート スーパーグラフィックツール

その1.慣れてしまうとマウスがいらない

DrawingSlate	￥74,800
Matier Ver2.0	￥39,800
合計定価	￥114,600

ツクモ特価￥ 89,800

その2.ハイクオリティなのにこんなに安い

BJC-600J	￥120,000
プリンターケーブル	￥ 4,800
Matier Ver2.0	￥ 39,800
合計定価	￥164,600

ツクモ特価￥116,000

大容量記憶装置

MO特選セット

SCSIボードが必要な場合にはセット価格に￥22,000加算となります

Logitec **LMO-200**(128MB) ￥79,800

SCSIケーブル
MOメディア
ターミネータ

ツクモ特価￥ 76,800

Logitec **LMO-400**(230MB) ￥158,000

MOメディア
SCSIケーブル
ターミネータ

ツクモ特価￥138,000

Panasonic **LF-3200B**(230MB) ￥168,000

SCSIケーブル
MOメディア

ツクモ特価￥138,000

COPAL **CS-M230PA**(230MB) ￥148,000

SCSIケーブル
MOメディア

ツクモ特価￥138,000

ハードディスク

240MBハードディスク..... ツクモ特価￥39,800～
290MBハードディスク..... ツクモ特価￥46,800～
350MBハードディスク..... ツクモ特価￥49,800～
520MBハードディスク..... ツクモ特価￥74,800～

CD－ROMドライブ

	ツクモ特価
CD－ROMドライバーソフト＋ケーブル	￥ 9,200

CD－ROMドライブ(2倍速)

		ツクモ特価
ELECOM	ECD-250(TOSHIBAドライブ)	￥42,800
メルコ	CDS-E(SONY/トレー)	￥24,800
Logitec	LCD-550-DV(TOSHIBAドライブ)	￥39,800
SONY	CDU-7811(SONYドライブ)	￥39,800
緑電子	CXA-301S(NECドライブ)	￥29,800
PIONEER	DR-U104X(4倍速)	￥69、800

多連装CD－ROMドライブ

PIONEER	ツクモ特価
DRM-602X(6連装2倍速) 台数限定	￥ 55,800
DRM-604X(6連装4倍速) <輸入品>	￥135,000

MIDIコンピュータミュージック特選セット

Roland SC-55mkIIセット

SC-55mkII	￥69,000
TS-6GM1	￥39,800
MIDI変換ケーブル	￥ 4,000
合計定価	￥112,800

ツクモ特価￥79,800

Roland SC-88セット

SC-88	￥89,800
TS-6GM1	￥39,800
MIDI変換ケーブル	￥ 4,000
合計定価	￥133,600

ツクモ特価￥99,000

簡単コンピュータミュージック

Music Card for X680x0 (TS-6GM1)
発売記念キャンペーン特価
￥24,800

・音源を搭載したMIDIボードの登場。これ1枚で手軽にMIDIコンピュータミュージックが楽しめます。
GM規格・MT－32・CM－64等の音色配列をサポート。最大同時発音数16。
ソフトウェア「Mu-1GSお試し版」付

パソコン通信

モデム

		ツクモ特価
AIWA	PV-AF144V5	￥29,800
OMRON	ME1414B	￥22,800
MicroCORE	MC14400FX	￥23,000
Panasonic	TO-703B	￥27,800
SUNTAC	MS144AVF	￥26,800

通信ソフト

SPS	た～みのる2	￥13,000
SHARP	Communication SX-68K	￥15,800

ソフトウェア

	ツクモ特価
OS-9/X68030 V2.4.5	￥20,000
Technical Tool Kit V.2.4.5	￥16,000
Ultra C & Professional Pack V1.1	￥36,000
X Windows V11.5	￥24,000
SX-WINDOW Ver3.1システムキット(NEW)	￥18,200
SX-WINDOWデスクアクセサリ集	￥11,800
C COMPILER Ver2.1 NEWKIT	￥35,800
Easydraw SX-68K	￥15,800
Easypaint SX-68K	￥10,200
SOUND SX-68K	￥12,600
Communication SX-68K	￥15,800
Matier Ver2.0	￥29,800
CD-ROM Driver	￥ 4,800
SX-PhotoGallery	￥15,800
DoubleBookin'	￥12,800
SX広辞苑(CD-ROM別)	￥17,800
EGWord SX-68K	￥47,800
SX-WINDOW開発キット	￥31,800
開発キット用ツール集	￥10,200
倉庫番リベンジSX-68K	￥ 5,400
MUSIC SX-68K	￥30,400
XDTP SX-68K	￥28,000
フォントデザインツール書家万流 SX	COMMING SOON
Super BUSINESS	NOW WAITING

秋葉原

至お茶の水
昌平橋通り
パソコン本店
5号店
ツクモソフト8号店
買取センター
万世橋
警察
万世店
三菱銀行
ラジオセンター店
中央通り
DOS/Vパソコン館
至神田
秋葉原駅
JR山手・京浜東北線
ニューセンター店
至浅草橋

名古屋

若宮大通り
矢場町
名古屋2号店
名古屋1号店

札幌

ツクモ札幌店
DEPO ツクモ2番街店

お支払い方法

あなたのご都合に合わせていろいろ選べます。

クレジット払い

月々¥3,000以上の均等払いも頭金なし。夏・冬ボーナス2回払いもOK！

カード払い

¥5,000以上
通信販売での御利用カード
ツクモグローバルカード・セントラル・ジャックス
※御本人様より電話で通信販売部へお申し込み下さい。

各種リース払い

詳しくは各店にご相談下さい。

現金書留払い

〒101-91 東京都千代田区神田郵便局私書箱135号
ツクモ通販センター Oh!X係

代金引き換え配達

お申し込みは電話1本でOK!
配達日の指定もできます。

銀行振込払い

事前にTELでお届け先をご連絡下さい。
三和銀行 秋葉原支店
(普) 1009939 ツクモデンキ

商品についてのお問い合わせは各店に

秋葉原

平日(営)AM10:45～PM7:30日・祝AM10:15～PM7:00

ツクモパソコン本店II 4F
03-3253-1899
03-3253-4199(代)
(休)木曜日

ツクモニューセンター店
03-3251-0987
(休)木曜日

名古屋

(営)AM10:00～PM7:00

ツクモ名古屋1号店
052-263-1655
(休)木曜日

ツクモ名古屋2号店
052-251-3399
(休)水曜日

札幌

(営)AM10:30～PM7:30

ツクモ札幌店
011-241-2299
(休)木曜日

DEPOツクモ2番街店
011-242-3199
(休)木曜日

★商品はお電話受け付けより、標準日数3日～1週間でお届け致します。(一部地域を除く)
★表示価格には消費税は含まれておりません。

安いのに親切
TSUKUMO
九十九電機株式会社

マイコン専門ショップ

P&A

SHARPエキスパートショップ

今が購入のチャンス!

SHARP

8/18~9/17

注目!!冬のボーナス一括払い手数料(金利)無料(平成6年9月末/10月末/11月末/12月末のいずれかをご指定下さい。)

X68030お買い得セット

(クレジット表:送料・消費税込み)

①X68030

●CZ-500C
●CZ-607D-TN(0.31mm、チューナー付)

定価合計¥497,800
P&A超特価¥299,000

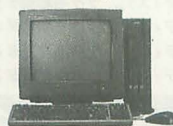
12回	27,700	24回	14,400	36回	10,000	48回	7,800	60回	6,500

②X68030 HD

●CZ-510C
●CZ-607D-TN(0.31mm、チューナー付)

定価合計¥587,800
P&A超特価¥398,000

12回	36,300	24回	19,200	36回	13,300	48回	10,400	60回	8,700

③X68030 Compact

●CZ-300C
●CZ-607D-TN(0.31mm、チューナー付)

定価合計¥487,800
P&A超特価¥328,000

12回	30,000	24回	15,800	36回	11,000	48回	8,600	60回	7,200

④X68030 Compact HD

●CZ-310C
●CZ-607D-TN(0.31mm、チューナー付)

定価合計¥577,800
P&A超特価¥393,000

12回	35,900	24回	18,900	36回	13,100	48回	10,200	60回	8,600

■モニター変更の場合
●CZ-608D………………に変更の場合¥ 3,000
●CZ-615D(チューナー付)に変更の場合¥56,000
●CZ-621D(B)…………に変更の場合¥64,000
}加算して下さい。

X68000 Compact XVI

旧シリーズ今が買いどき!!
(クレジット表:送料・消費税込み)送料¥2,000・消費税別

①本体+モニター

●CZ-674C-H
●CZ-608D-H

定価¥392,800
P&A超特価¥147,000

12回	13,400	24回	7,100	36回	4,900	48回	3,800	60回	3,200

②本体+モニター+FDD(5"×2)

●CZ-674C-H
●CZ-608D-H
●CZ-6FD5(FDD)

定価¥492,600
P&A超特価¥195,000

12回	17,700	24回	9,300	36回	6,500	48回	5,000	60回	4,200

③本体+モニター(TVチューナー付)

●CZ-674C-H
●CZ-607D-TN
●RGBケーブル

定価¥397,800
P&A超特価¥144,000

12回	13,200	24回	6,900	36回	4,800	48回	3,700	60回	3,100

④本体+モニター(TVチューナー付)+FDD(5"×2)

●CZ-674C-H
●CZ-607D-TN
●RGBケーブル
●CZ-6FD5(FDD)

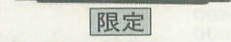
定価¥497,600
P&A超特価¥192,000

12回	17,500	24回	9,200	36回	6,400	48回	5,000	60回	4,200

■モニターの変更 ※③、④のモニターを
●CZ-615D(チューナー付)に変更の場合¥56,000
●CZ-621D(B) に変更の場合¥64,000
}加算して下さい。

X68000 Compact XVI

本体(単品)
⊙CZ-674C
定価¥298,000
P&A超特価
¥85,000

カラーイメージユニット

限定
●CZ-6VT1-BK
定価¥69,800
特価¥52,500

インテリジェントコントローラ

限定
●CZ-8NJ2
定価¥23,800
特価¥13,800

X68000/68030用 メモリボード

(送料¥700・消費税別)

■I/Oデータ
●SH-5BE4-8M(30用)……特価¥39,500
●SH-6BE1-1ME(600C用)…特価¥10,200
●PIO-6BE1-AE(ACE/PRO /PROII用)特価¥10,200
●PIO-6BE2-2ME(拡張スロット用)特価¥21,000
●PIO-6BE4-4ME(〃)特価¥35,300

■シャープ
●CZ-5BE4(30用)…………特価¥39,800
●CZ-5ME4(5BE4用増設)特価¥36,500
●CZ-6BE2A(XVI用)……特価¥38,900
●CZ-6BE2B(XVI、674C増設)特価¥37,500
●CZ-6BE2D(674C用)…特価¥20,500

モデム&FAXモデム

(送料¥1,000)

〈インテグラン〉
●MP1414F(FAXモデム・ポケット型)…………特価¥31,000
〈サン電子〉
●MS1414AVF(FAXモデム・ボックス型)……特価¥30,000
〈アイワ〉
●PV-AF24V5(FAXモデム・ボックス型)……特価¥22,500
●PV-PF144(FAXモデム・ポケット型)………特価¥32,000
●PV-AF144V5(FAXモデム・ボックス型)……特価¥36,000
〈オムロン〉
●MD-96XT10V (FAXモデム・ボックス型)……特価¥30,000
●MD-144XT10V(FAXモデム・ボックス型)……特価¥35,000
〈マイクロコア〉
●MC14400FX(W)(FAXモデム・ボックス型)…特価¥23,000
●MC24FC5(W) (FAXモデム・ポケット型)…特価¥20,000

MO&CD-ROM

(送料¥1,000)

■CS-M120(コパル)
光磁気ディスク(X68000用)
●ケーブル、ターミネータ付
定価¥178,000
特価¥93,000

■LMO-FMX330TS
(ロジテック)●ケーブル付
定価¥168,000
特価¥97,000

MO (ケーブル別売)
●UL-312E-S(緑電子)…特価¥ 62,000
●MO-120S(ICM)………特価¥ 88,000
●MO-230S(〃)………特価¥110,000
●LMO-340(ロジテック)…特価¥ 85,000
●LMO-400(〃)…特価¥110,000

CD-ROM
●CXA-301S(緑電子)…特価¥ 25,500
●CXA-450S(〃)…特価¥ 59,400
●CD-310S(ICM)………特価¥ 33,800
●CD-450N(〃)………特価¥ 63,600
●CD-600S(〃)………特価¥ 68,600
●LCD-550(ロジテック)…特価¥ 38,800

東京システムリサーチ製(X SIMM)

(送料¥700・消費税別)

(X SIMM VI)
⊙XVIシリーズ専用SIMM増設式メモリボード
●X SIMM VI(634C用)…定価¥16,500➡特価¥13,000
●X SIMM VIc(674C用)・定価¥16,500➡特価¥13,000
⊙増設SIMMメモリ(72PIN)
●4MB(70ns)…………………特価¥13,500
●8MB(70ns)…………………特価¥29,000
●4MB(60ns、24MHz以上用)………特価¥17,000
●8MB(60ns、24MHz以上用)………特価¥31,000

●6MB(70ns、メーカー純正品)……特価¥31,000
(X SIMM 10)⊙SIMM増設式メモリボード
●X SIMM 10………定価¥18,000➡特価¥15,700
⊙増設SIMMメモリ ●1MB×2……特価¥ 9,000
●4MB×2……特価¥30,000
●10MB例 X SIMM10+1MB×2+4MB×2‥¥54,700

X68000/68030専用ハードディスク

(送料¥1,000・消費税別)

外付

■ロジテック
⊙SHD-B240N-FMX(ケーブル付)(240MB、14ms、64K)
…………………………定価¥59,800▶特価¥45,000
⊙SHD-B340N-FMX(ケーブル付)(340MB、12ms、128K)
…………………………定価¥74,800▶特価¥52,000

■富士通
⊙HD-M260(モッキンバード)(260MB、14ms、256K)
…………………………特価¥39,800
⊙HD-K520(モッキンバード)(520MB、12ms、240K)
…………………………定価¥128,000▶特価¥69,800

■ジェフ
⊙GF-270(270MB、12ms、128K)
…………………………定価¥89,800▶特価¥59,000
⊙GF-540(540MB、12ms、128K)
…………………………定価¥128,000▶特価¥69,800

内蔵

■CZ-500C/300C専用
⊙CZ-5H08(80MB/23ms)
…………………………定価¥ 98,000▶特価¥71,800
⊙CZ-5H16(160MB/18ms)
…………………………定価¥135,000▶特価¥99,500

ズバリご奉仕

P&Aならではの 新品パソコン5年保証

《業界No.1の"P&Aメンテナンスサポート"》
最高の保証システム
① 業界最長の新品パソコン5年保証
（※モニター・プリンター3年間保証！ ※一部商品は除きます。）
② 中古パソコンの1年間保証（※モニター・プリンター6ヶ月間保証！）
③ 初期不良交換期間3ヶ月（※新品商品に限らせていただきます。）
④ 永久買取保証
⑤ 配達日の指定OK！（土曜・日曜・祭日もOK！）
⑥ 夜間配達もOK！（※PM6:00〜PM8:00の間 ※一部地域は除きます。）

便利でお得な支払いシステム
① 翌月一括払い手数料無料（ご利用下さい。）
② 業界No.1の低金利！
③ 月々の支払いは¥1,000より
④ 9ヶ月先からのスキップ払いOK！
⑤ 84回までの分割、ボーナス併用OK！
⑥ カレッジクレジット
⑦ ステップアップクレジット
⑧ ボーナスだけで10回払いOK！
⑨ 現金一括支払いOK！
⑩ 商品到着払いOK！（代引き手数料が必要になります。10万円まで900円）
（※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。）

●法人向け リースシステム
業務に最適なシステムを構築します。
損金処理が可能なリース契約をどうぞ。

周辺機器コーナー （送料¥1,000・消費税別）

カラーイメージスキャナ
■JX-325X 限定
定価¥190,000
特価**¥79,800**

カラーイメージジェット

■IO-735X-B
定価¥248,000
特価**¥128,000**

ビデオスキャナー
■CZ-6VS1
定価¥178,000
特価**¥135,000**

FDD（5インチ×2基）
■CZ-6FD5
定価¥99,800
P&A超特価
¥49,800

プリンター（ケーブル用紙付）
- MJ-500V2 （エプソン）…特価¥44,300
- MJ-1000V2（ 〃 ）…特価¥64,300
- MJ-700V2C（ 〃 ）…特価¥78,300
- BJ-220JC （キャノン）…特価¥58,000
- BJ-10V Lite（ 〃 ）…特価¥31,300
- BJ-15V PRO（ 〃 ）…特価¥39,700
- LBP-A404GII（ 〃 ）…特価¥99,500
- BJC-600J （ 〃 ）…特価¥78,300
- JET505J PLUS（YHP）…特価¥50,300

光磁気ディスク（X68000用）

■CS-M120（コパル）
●ケーブル、ターミネータ付
¥178,000
特価**¥93,000**

- CZ-6BV1………定価¥21,000▶特価¥15,900
- CZ-8NM3………定価¥ 9,800▶特価¥ 7,200
- SH-6BF1………定価¥49,800▶特価¥36,500
- CZ-6BP1………定価¥79,800▶特価¥57,000
- CZ-6BS1………定価¥29,800▶特価¥21,500
- CZ-8NJ2（限定）…定価¥23,800▶特価¥13,800
- CZ-6CS1（674C用）・定価¥12,000▶特価¥ 8,900
- CZ-6CR1（RGBケーブル）・定価¥ 4,500▶特価¥ 3,600
- CZ6CT1（テレビコントロール）・定価¥ 5,500▶特価¥ 4,400
- CZ-6BP2………定価¥45,800▶特価¥33,300
- CZ-5MP1（X68030用）・定価¥54,800▶特価¥42,000

送料¥700・消費税別
■システム サコムボード
- SX-68MII（MIDI）
定価¥19,800
特価¥13,500
- SX-68SC（SCSI）
定価¥26,800
特価¥17,500

X68000用ソフトコーナー （送料¥700・消費税別）

- Z's STAFF PRO68K Ver.3.0（ツァイト）……定価¥58,000▶特価¥37,500
- Z's TRIPHONYデジタルクラフト（ツァイト）……定価¥39,800▶特価¥27,000
- マジックパレット（ミュージカルプラン）……定価¥19,800▶特価¥14,200
- たーみのる2（SPS）……定価¥17,800▶特価¥13,000
- サイクロンEXPRESS α68……定価¥98,000▶特価¥69,000
- Video PC for X680X0（マイクロウェアシステムズ）……定価¥58,000▶特価¥46,400
- X WINDOWS V.11.5（マイクロウェアシステムズ）……定価¥30,000▶特価¥25,500
- Double Book IN（計測技研）……定価¥12,800▶特価¥ 9,600
- OS-9/X68030 V.2.4.5（マイクロウェアシステムズ）……定価¥25,000▶特価¥19,900
- C&Professional Pack V.3.2（マイクロウェアジャパン）……定価¥80,000▶特価¥57,800
- マチエール Ver.2.0……定価¥39,800▶特価¥28,800
- F-Calc for X68K……定価¥14,800▶特価¥11,000
- CZ-214MSD SOUND PRO68K……定価¥15,800▶特価¥11,300
- CZ-215MSD Sampling PRO68K……定価¥17,800▶特価¥12,500
- CZ-225BSV Multiword Ver.2.0……定価¥32,000▶特価¥23,000
- CZ-227BS TOP財務会計PRO-68K……定価¥200,000▶特価¥154,000
- CZ-243BSD CYBERNOTE PRO68K……定価¥19,800▶特価¥15,000
- CZ-247MSD MUSIC PRO68K（MIDI）……定価¥28,800▶特価¥20,500
- CZ-249GSD CANVAS PRO68K……定価¥29,800▶特価¥22,000
- CZ-251BSD Hyperword……定価¥39,800▶特価¥29,400
- CZ-253BSD CARD PRO68K Ver.2.0……定価¥29,800▶特価¥22,700
- CZ-257CSD Communication PRO68K Ver.2.0……定価¥19,800▶特価¥15,300
- CZ-261MSD MUSICstudio PRO68K Ver.2.0……定価¥28,800▶特価¥21,200
- CZ-263GWD Easypaint SX-68K……定価¥12,800▶特価¥ 9,800
- CZ-264GWD Easydraw SX-68K……定価¥19,800▶特価¥15,300
- CZ-265HSD NewPrint Shop Ver.2.0……定価¥20,000▶特価¥15,400
- CZ-266BSD PressConductor PRO68K……定価¥28,800▶特価¥22,000
- CZ-267BSD CHART PRO68K……定価¥38,000▶特価¥29,800
- CZ-271BWD EG-Word……定価¥59,800▶特価¥44,900
- CZ-272CWD Communication SX68K……定価¥19,800▶特価¥14,500
- CZ-274 MWD MUSIC SX68……定価¥38,000▶特価¥29,300
- CZ-275MWD SOUND SX68K……定価¥15,800▶特価¥11,500
- CZ-284SSD OS-9/X68000 Ver.2.4……定価¥35,800▶特価¥25,600
- CZ-286BSD BUSINESS PRO68K……定価¥28,000▶特価¥20,500
- CZ-288LWD開発キット（workroom）……定価¥39,800▶特価¥29,700
- CZ-289TWD 開発キット用ツール集……定価¥12,800▶特価¥ 9,600
- CZ-290TWD SX-WINDOW ディスクアクセサリー集……定価¥14,800▶特価¥11,500
- CZ-295LSD C-Compiler PRO68K Ver.2.1 NEW KIT……定価¥44,800▶特価¥32,500
- CZ-296SS/SSC SX-WINDOWS Ver.3.1……定価¥22,800▶特価¥17,600

☆ゲームソフト25%OFF OK！（一部ソフト除く）

全国通販 ★頭金なし！★即日発送

●お近くの方はお立寄り下さい。専門係員が説明いたします。
●本体単品で特価で受付します。詳しくは電話にてお問合せ下さい。
●ビジネスソフト定価の20%引きOK！TEL下さい。

P&A特選 今月の中古特選品

- CZ-600C‥¥45,000
- CZ-601C‥¥45,000
- CZ-611C‥¥50,000
- CZ-652C‥¥55,000
- CZ-612C‥¥75,000
- CZ-603C‥¥65,000
- CZ-653C‥¥58,000
- CZ-612C‥¥70,000
- CZ-623C‥¥70,000
- CZ-674C‥¥70,000
- CZ-634C‥¥100,000
- CZ-644C‥¥145,000

※上記は単品価格、モニター別売。

新古品 限定
●CZ-674CH
●CZ-608DH
¥128,000

中古品
●CZ-674CH
●68000専用モニター付
¥89,000

限定
●CZ-634CTN（チタン）（中古）
●CZ-613D（グレー）（新品）
¥180,000
（モニターをCZ-614CTN（チタン）に変更の場合¥20,000加算）

中古品
●CZ-623C-TN
●68000専用モニター
¥98,000

新古品 限定
●CZ-644CTN
●CZ-604DB
¥208,000

中古品
●CZ-644CTN
●68000専用モニター付
¥178,000

高額買取り（新品もOK）格安販売

■まずはお電話下さい。
下取り専用 買取り電話▶ ☎03-3651-1884 FAX. 03-3651-0141
■下取り・買取りで、お急ぎの方は、直接当社に来店、または宅急便にてお送りください。

買取り価格…完動品・箱／マニュアル／付属品の価格です。中古販売…1年間保証付。

●下取りの場合…価格は常に変動していますので査定額を電話で確認してください。（差額は、P&A超低金利クレジットをご利用ください。）
●買取りの場合…現品が着き次第、3日以内に高価買取金額を連絡し、振込み、又は書留でお送り致します。
●近郊の方はP&A本店に直接お持ちください。即金にて¥5,000,000までお支払い致します。

●最新の在庫情報・価格はお電話にてお問い合せください。
●買い取りのみ、または、中古品どうしの交換も致します。詳しくは電話にて、お問い合せください。
●価格は変動する場合もございますので、ご注文の際には必ず在庫をご確認ください。
●本商品の掲載の商品の価格については、消費税は、含まれておりません。
●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せください。

P&A特選 パソコンラック&OAチェアー（消費税込み）（送料無料、離島を除く）

①3段**¥8,240** ②4段**¥9,785**

※全機種→キャスター付 ※フレーム色：ホワイト
※上から2番目棚板移動可能（4段）
※3段の場合、上から2番目の棚板は付いておりません。

③4段**¥12,875**（持ち帰り可能です ご来店下さい。）

※上下2分割式／スライドマウステーブル、中棚板は2段階に可動します。
※フレーム色：グレー

①**¥9,270**
●布張り ダークグレー
●ガスシリンダー

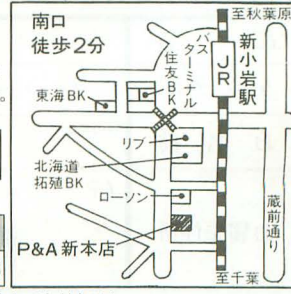
②**¥11,330**
●布張り ダークグレー
●ガスシリンダー
●肘付

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで現金書留でお送りください。（プリンター・フロッピーの場合、本体使用機種名を明記のこと）
〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。●1回〜84回払いまで出来ます。但し、1回のお支払い額は¥1,000円以上。
〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。（電信扱いでお振込み下さい。）

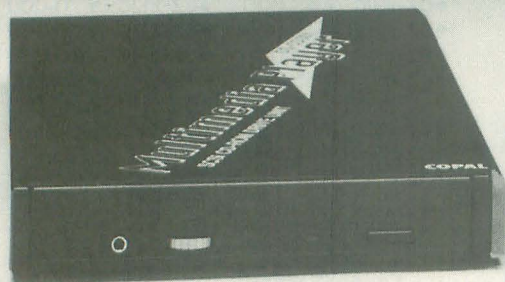
〔振込先〕さくら銀行 新小岩支店
当座預金 2408626（株）ピー・アンド・エー

超低金利クレジット率

回数	3	6	10	12	15	24	36	48	60	72
手数料	2.6	3.5	4.4	4.9	7.8	10.4	14.4	18.9	24.4	31.8

※お支払いは、便利な商品到着払い〈手数料（10万円まで900円）要〉をご利用下さい。

株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目2番地20号
●営業時間：AM10：00〜PM7：00 日・祭：AM10：00〜PM6：00
☎03-3651-0148（代）
●定休日／毎週水曜日
FAX.03-3651-0141　MAC/DOS Vフロア ☎03-3655-4454

●価格は変動します。ご注文の際は必ずお電話で価格と在庫をご確認下さい。●本広告に掲載の商品には送料及び消費税は含まれておりません。

X680x0にジャストフィット
精悍な黒モデル フルラインナップ

エアフィルタ交換不要の3.5インチ光磁気ディスクユニット

CS-M120PX

定価¥178,000　**通販特価¥108,000**

■平均シークタイム30ms,回転数3600rpm,記憶容量128MBの高性能ドライブ。
■今回お買い求めの方に限りケーブル・ターミネーターをサービス。
＊X68000,Human68Kでのご使用となります。SX-WINDOWでのご使用についてはお問いあわせください。

外付ハードディスクユニット

CS-H540X

定価¥128,000　**通販特価¥78,000**

■フォーマット容量540MB,平均アクセスタイム12ms,ターミネータ付,ケーブルはサービス

CS-H240X

定価¥79,800　**通販特価¥44,000**

■フォーマット容量240MB,平均アクセスタイム15ms,ターミネータ付,ケーブルはサービス

デバイスドライバー付倍速CD-ROMユニット

CS-CD301X

定価¥59.800　**通販特価¥38,000**

■各種フォーマット対応 CD-DA,XA,Photo-CD,CD-Bridge,CD-1フォーマット対応
■キャディのいらないトレー式、ケーブル/ターミネータ標準添付(ディジーチェーン接続が可能)

＊4機種ともSCSI I/Fボードはパソコン本体に附属のものまたは純正品が使用可能です。その他サードパーティ製のSCSI I/Fボードとの接続についてはお問い合わせください。
＊ご注文の際にはご希望のケーブルをご指定下さい。
(CS-H540X、CS-H240Xについては、ユニット側はフルピッチコネクタで、その他の機種はハーフピッチコネクタです。)

●製品についての情報は、FAXステーションから次の要領で取り出して下さい。　FAX Station

1. FAXの受話器をあげて
2. FAXステーション(☎03-3499-0177)にダイヤルして下さい。
3. 音声案内に従って(ダイヤル回線の方はピポパのトーン信号に切り換えて)＃を押します。
4. 音声案内に従って情報番号6200＃を押し、最後に終了の＃を押します。
5. 送受信のメッセージ終了後(約3秒後ピー音を確認)ファクシミリのスタートボタンを押して受話器を戻します。→「製品情報」をお受取下さい。

●お申し込みは、注文書の太枠線内にご記入の上FAXまたは郵送にてお送り下さい。

●お申し込み先　コパル綜合サービス株式会社　通販係
〒174　東京都板橋区志村2-16-20
TEL.03-3965-1144　FAX.03-3968-1029

*商品の技術的なご質問·ご相談はユーザーサポート係まで
TEL.03-3965-1161

●お支払いは銀行振込で、下記口座までお振込下さい。
(振込手数料はお客様負担で電信扱いでお振込下さい)

口座番号　第一勧業銀行　志村支店　普通預金　No.1369382
口座名義　コパル綜合サービス株式会社

●商品の引渡しは代金お支払い後となります。
●商品はご入金後、原則として3日以内に発送します。
(在庫切れの場合は、ご連絡いたします。)

■ご注文書　FAX 03-3968-1029

品　名		ご注文台数　台	ご連絡先 TEL.(　) FAX.(　)
ケーブル※1	□フル～ハーフ □ハーフ～ハーフ □フル～フル		
お名前	ふりがな		
お届先住所	(〒　-　) 都道府県 区市郡		1.会社　2.自宅

弊社記入欄
受付番号
受付日
納入日
備考

※1 ご希望のケーブルをご指定ください。

オリジナルアプリケーション開発速報　1994年9月1日発行（毎月1回1日発行）通巻12号 *FirstClassTechnology* 発行

緊急特価

X68030(CZ-500C)+040turboバンドルセット ¥328,000!

新発売その1

SX-WINDOW用CD-ROM辞書検索ソフト

SX広辞苑《EPWING対応版》

標準価格　ソフト単体 **¥19,800**

岩波書店「広辞苑第4版CD-ROM版」バンドルセット **¥43,800**

あの、SX広辞苑がグレードアップして新登場！

岩波書店から発売されている辞書CD-ROM「広辞苑第4版CD-ROM版」。これは、同社の国語辞典「広辞苑第4版」の内容をそのままCD-ROM化したもので、文章や挿絵もすべて収録されています。また、電子メディアの特長をいかして、鳥の鳴き声の音声データや色見本情報なども収録。

SX広辞苑《EPWING対応版》は、この「広辞苑第4版CD-ROM版」を効率的に検索し、120%活用するためのソフトです。

●SX広辞苑《EPWING対応版》の特長

・豊富でパワフルな検索方法により、必要な情報をすばやくピックアップ。

・使う側に立って操作系をリニューアル。さらに簡単に、さらに鋭く作業を行なえます。

・広辞苑の最新版である第4版をもとにしたCD-ROMを使用するので、よりコンテンポラリーなキーワードにアクセス可能です。

・SX-WINDOW上で動作するので記事の参照や引用がとても簡単。シャーペンやEGWordと組み合わせて活用できます。（ただし、広辞苑では大量の引用は禁止されています）

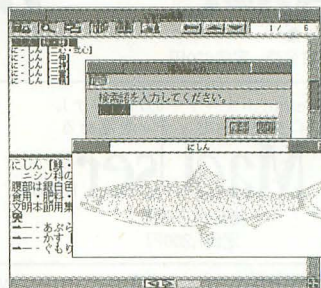
・シャーペンと融合して語句の検索を行なうシャーペン用外部コマンド"LightWing.X"を同梱。複雑な検索を行なう場合はSX広辞苑.Xを、普段よく使う単純な検索にはLightWing.Xを、という使い分けも可能です。

・広辞苑第4版CD-ROM版と同様に、EPWING(V1)規約にもとづいたCD-ROMタイトルなら、ほとんどのCD-ROMの内容を検索できます。

●当社で動作を確認したEPWING(V1)タイトル

現在約20タイトル発売されているEPWING(V1)準拠のCD-ROMのうち、以下のタイトルについては当社で動作を確認しました。なお、SX広辞苑《EPWING対応版》上での動作に関して、各タイトルの出版社に問い合わせることはご遠慮ください。

タイトル	出版社
・広辞苑第4版CD-ROM版	岩波書店
・岩波電子日本総合年表[EPWING版]	岩波書店
・CD-ROM最新医学大辞典[スタンダード版]	医歯薬出版
・漢和辞典漢字源[EPWING版]	学習研究社
・リーダース英和辞典	研究社
・三省堂ワードハンター　―マルチROM辞典―	三省堂

この他のタイトルについても動作確認作業を進めています。

●動作環境

・SX-WINDOW Ver3.0以上

・SX-WINDOW動作中の空きメモリとして1MB以上を推奨

・CD-ROMドライブ(CD-ROM Driver Ver2.0が付属するので、CD-ROM Driverを別途お買い上げいただく必要はありません。CD-ROM Driverのマニュアルや添付ソフト等は付属しません)

新発売その2

X680x0用Ether net接続パック

EthernetStarterPack/X680x0

発売記念特価 **¥78,000**

ESP/Xは、Ether netアダプタ「Ether+」と、TCP/IPドライバ、そして基本的なアプリケーションからなるパッケージです。

・Ether+(米コンパチブルシステムズ社製)
　SCSIインターフェースを介してEther netとX680x0を接続するためのハードウェアです。
　※10BASE-2対応モデル・10BASE-T対応モデルの2種類があります。

・TCP/IPドライバ
　X680x0でTCP/IPをサポートするドライバ。ソケットも利用可能です。

・基本的なアプリケーション
　ftp、telnet(いずれもクライアント)等、基本的なアプリケーションを標準添付。ドライバを活用するためのライブラリも付属します。

●動作環境

・Human68k ver3.0以上

・メモリ常駐量500KB前後

・SCSIインターフェース内蔵機種以外はSCSIボードが必要

X68030用　68040搭載アクセラレータ

X68040turbo

標準価格 **¥98,000**　ヒートシンク別売 **¥1,000**

040turboは、68040を搭載したX68030(5インチタイプ)専用のアクセラレータです。040turboを装着することで得られるパフォーマンスは、従来の2～3倍！ 計算、特に浮動小数点演算中心のソフトならば、さらにそれ以上の高速化も望めます。

詳しくはソフトバンク刊「X68040turbo～A Story of Making "After X68030"～」(BEEPs著)をご覧ください。

040turboは当社のショップBASIC-HOUSEでの直販、および通販でのみお買い求めいただけます。ご注文いただいてからしばらくお待ちいただく場合もありますので、お早めにご注文ください。

最新ベンチマーク(当社調べ)

プログラム	68030 (キャッシュON)	68040 (キャッシュOFF)	68040 (コピーバックキャッシュ)
pv.r	5.4863	4.8292	18.5058
ｷｬﾝﾊﾞｽ.x*1	16.88	18.45	5.92
dhry.x*2	5903	3888.0	22624.4

*1　SX-WINDOW上で「草原.JPG」が表示されるまでの秒数

*2　Dhrystone Benchmark, Version 2.1

発売中　SX-WINDOW用スケジュール管理ソフト

DoubleBookin'

標準価格 **¥12,800**

発売中　バージョンアップ!!

CD-ROM Driver Ver2.00

標準価格 **¥4,800**

発売中　SX-WINDOW用Photo-CDビュアー

SX-PhotoGallery

標準価格 **¥15,800**　通販特価 **¥15,000**

発売中　X680x0用フリーソフトウェア集CD-ROM

FreeSoftwareSelection Vol.2

標準価格 **¥6,000**

お求めはお近くのパソコンショップ、または弊社通販部(TEL:0286-22-9811)へお申し込みください。
通販ご希望の方は、ソフト代金+送料¥1,000に消費税を加え、ご住所・お名前・電話番号・商品名を明記した紙を同封の上、現金封筒でお申し込みください。

低金利クレジット　通信販売送料 全国一律¥1,000 長期クレジット可能　※表示価格に消費税は含まれておりません

株式会社 計測技研　マイコンショップ BASIC HOUSE 本社／ショールーム／通販部
〒321 栃木県宇都宮市竹林町503-1
TEL 0286-22-9811　FAX 0286-25-3970

当社製品に関する最新情報や詳しい資料は、当社サポートネットTECOSYS-3(0286-51-1430/9600bps)でもご覧いただけます。どうぞご利用ください。

ソフトバンクの14大雑誌

SOFT BANK

ハード・ソフト 活用情報を満載
NEC PC-98活用誌

毎月1,15日発売 定価620円

X68000、X1、MZユーザーのための情報誌

Oh!X

毎月18日発売
定価600円

富士通FMシリーズ情報誌
for FMTOWNS/MARTY/R/V etc

Oh!FM TOWNS

毎月18日発売・定価620円

パーソナルコンピュータ総合情報誌

月刊PC

毎月18日発売
定価650円

Macintoshユーザーのパーソナル
プロダクティビティを高める

MacUser 日本語版

毎月18日発売
定価1,200円

企業ユーザーのPC&WS活用を
支援する情報誌

PCWEEK 日本語版

毎週金曜日発行・年間12,000円

C言語技術情報誌

C MAGAZINE

毎月18日発売・定価1,000円

ネットワークコンピューティングを
推進する実務マガジン

LAN TIMES 日本語版

毎月8日発売・定価1,480円

Windowsと
GUI環境を活かす専門誌

THE WINDOWS

毎月8日発売・定価980円

アプリケーション指向のUNIX活用誌

UNIX USER

毎月8日発売・定価1,280円

IBM PCと互換機ユーザーの総合誌

DOS/V magazine

毎月8日発売・定価780円

コンピュータ技術者必携
第2種・第1種・オンライン試験

月刊情報処理試験

毎月8日発売・定価780円

スーパーファミコン100%

The スーパーファミコン

隔週金曜日発売・定価390円

メガドライブの最強情報誌

BEEP! POWERFUL MEGA-MAGAZINE MEGADRIVE

毎月8日発売・定価540円

定価は税込み

お近くの書店でお求めください。

昨年に増して嬉しき残暑かな、ジャストのX68kペリフェラル

今年はマジに暑いっすね、でもこれが載る頃いきなり寒くなってたりして（笑）。
さて、今回はケーススタディです。

Example 1
「ってさー、いま初代機使ってんだけどー、やっぱ030欲しいんじゃん。でも俺ってお金ないしー、今度買うならDOS/Vとかいわれちゃったりすると立場無くなっちゃうしー、とりあえずゲームすんなら困んないしー、でもアーカイバとか遅くてたまんないしー、難しいことわかんないしー（以下略）」
そんな方には

▼MPUアクセラレーター H.A.R.P for MC68000
型番：DCMA00D1　対応機種：X68000初代.ACE.EXPERT.PRO.SUPER
定価：¥29,800（税別）
＊H.A.R.P for MC68000が貴方の悩みを解決します。既存のMPUと交換するだけであっという間に倍速動作、ソフトウェア上のパッチ等も不要です。手軽なインストレーションと優れたコストパフォーマンスが売りです。お客様は買いですよー。そして、もうひとつ、

▼拡張SIMMメモリーボード ER10S
型番：ER10Sn(SIMM未実装) 定価：¥14,800(税別)／ER10SDn（SIMM4MB1枚実装済）定価：¥39,800（税別）対応機種：X680x0全機種
＊安価なIBM PC用72ピンSIMMを採用した拡張I/Oスロット用メモリーボードことER10S、実装最大10MB、さらにH.A.R.P実装時の独自のメモリーサイクルモードにより、拡張スロット実装タイプのハンディを克服した高速なメモリーアクセスも可能としています。H.A.R.Pと一緒に導入すれば効果倍増！、SIMM無しと4MB×1枚実装の２モデルが選択できます。っつー訳でひとつまとめて買って下さいなっ。

Example2
神奈川県在住のＢさんは今年26才、高かった自動車保険も26才未満不担保の安い料率に切り替えられると喜んでいる脳天気なサラリーマンです。彼の仕事場にはなぜかUNIXのワークステーションがゴロゴロしていて、彼自信も趣味と実益をかねてスーパーユーザーをやっています。さて、Oh!Xとユニマガ（笑）を定期購読しているＢさん、98ともうすぐローンの終わるＸ68030を自宅に抱えてのいま一番の感心事はジャストのH.A.R.P－ＦＸ、030の50MHzなんて米軍のミサイル位でしか聞いたことがないし、仕事場でネームサーバーとメールサーバーとｆｔｐサイトを兼ねているかわいそうな030ワークステーションの事も気になるし、やっぱり速いにこしたことはないし、もしかしたら全部いっぺんに解決できるのではないかと考えていました。実は切れ者かもしれないＢさん、とりあえず26才になったら318iSのＭＴを買おうと思ってます。車両保険安くなるし。
そんな方には

▼MPUアクセラレーター H.A.R.P-FX（H.A.R.P for MC68030）
型番：DCMA30E1　対応機種：MC68030.PGAソケットの採用されたファームウェア（供給クロック25ＭＨｚ以下）
予価：¥68,030（税別）'94年８月出荷予定：予約受付中
＊MC68030互換MPUアクセラレーターことH.A.R.P-FX、たとえばX68030に実装した時には、元のクロックスピード25MHzを２倍し、オンボード上のMC68030RC50がフルバリー50MHz動作、さらにMPUオンチップのキャッシュメモリーがクロックスピードと相乗し優れたパフォーマンスを発揮します。もちろん、ソフトウェアの互換性を完全に維持、既存の環境で動作していたソフトウェアならまずOKです。X68030に限らず、68030を採用するパーソナルコンピューター、ワークステーションのほとんどに適応します。Ｂさんの仕事場でも、もちろん自宅でも活躍できるH.A.R.P-FX、車体400万弱のドイツ車買う前に、68,030円で買える幸せ。是非ともご用命を。

Example4
「ハコ、ハコが欲しいんだよー、ハコが！、判ってんの？、ハコだよハコ！ハマコーじゃないぞ、ハ・コ！。」
そんな方には

▼拡張I/Oスロット ESX68
型番：ESX68L4　対応機種：X680x0全機種
定価：¥39,800（税別）'94年７月出荷開始
＊なんだかよくわかりませんが、ハコと言われれば弊社はこれしかありません（笑）。特にマンハッタンタイプのユーザーにとって切実な問題である拡張I/Oスロットの不足を一気に解消できるものと確信しておりますこの製品、ESX68でございます。マンハッタンタイプでもいきなりPROシリーズと同じ4スロットが利用可能、高速バッファ搭載のインターフェースガードと外部スロット専用スイッチング電源により安定した動作を確保しています。これだけの装備でこの価格。選択の余地は無いに等しいはずです（笑）。

次回予告
製品出荷の遅れを取り戻し、いよいよ世界征服に向けて進み続けるジャスト他1社。ペリフェラルの開発もさることながら、システム事業部でついにWindows向けソフトウェアをリリースすることを画策、本格的な夏を迎え、アサヒの烏龍茶から特売品の水だし麦茶に切り替えた開発スタッフは、着々とプロジェクトは進行させつつあった。
彼らの新たな世界戦略とは如何に？、以下次号です。

※表示の定価は全て消費税別となっております。

※Motorolaはモトローラ社の登録商標、その他製品の名称等は一般に各メーカーの商標・登録商標です。

サポート　(有)エヌ・エム・アイ
開発・販売　(株)ジャスト

JAST

〒156 東京都世田谷区宮坂3-10-7 YMTビル3F
Phone.03-3706-9766 FAX.03-3706-9761 BBS.03-3706-7134

クイーン・オブ・デュエリスト外伝α+

X68000

※このソフトは18歳未満の方はご購入できません。

¥9,800（税別）

企画・開発／アグミックス
X68000/X68030対応

QD vs 外伝ファン待望、夢の対決だ！

★マンガ家8人が登場キャラクターをそれぞれデザイン！
★総勢10人のキャラクターを自由に選択！
★QDキャラ（美由紀、エミリー、龍凰）がマイナーチェンジして外伝キャラと対決！
★QD＆外伝キャラクターの18禁スペシャルグラフィックで満足度◎！
★キャラクターパターンはさらに増え、よりリアルな動きを再現！
★背景もX68000用にすべて描きかえました！
★PCMもさらにパワーアップして、迫力のサウンドを保証！
★必殺技、連続技、キャンセル技等の要素も含み、戦闘バランスもパワーアップ！

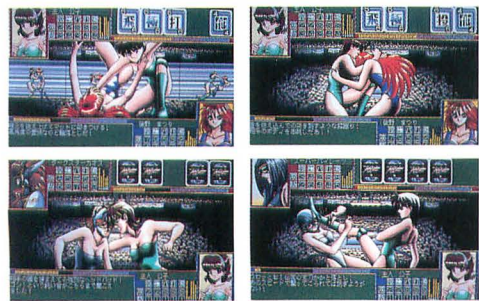
※画面写真はFM-TOWNS版です

目指すは最強団体！

X68000

レッスルエンジェルス3

新鋭達を集め、大器を育てて女子プロマットを制覇しろ!!

ゲーム業界初のプロレス団体経営シミュレーション！
新人、現役等約50名のレスラーが登場。
彼女の内、誰を引き込み、敵に回すかはプレイヤー次第。
最大3人までのマルチプレイが可能となり、そしてセーブデータ同士の団体対抗戦も加え、更に面白さが倍増した！
もちろん、ハイスピードなカードバトルも健在！
合体技等を加えた技はオールリニューアル。
これであなたもプロレス異次元ワールドに突入だ！

企画・開発／グレイト
X68000/X68030対応

¥7,800（税別）

株式会社エクシング TAKERU事務局
〒467 名古屋市瑞穂区苗代町2番1号
ブラザー技術開発センタービル2F
TEL(052)824-2493（受付時間：月～金 13:00～18:00）

営業所
東京営業所 (03) 5443-4967
大阪営業所 (06) 258-3024

T1002179090683 雑誌 02179-9